University of Alberta

VIDEO FOR LOW-BANDWIDTH TELENAVIGATION

by

W. Jonathan Baldwin ©

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.

Department of Computing Science

Edmonton, Alberta
Fall 2000

National Library of Canada
Acquisitions and Bibliographic Services
395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale du Canada
Acquisitions et services bibliographiques
395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-59562-5

Canada

University of Alberta

Library Release Form

Name of Author: W. Jonathan Baldwin

Title of Thesis: Video for Low-Bandwidth Telenavigation

Degree: Doctor of Philosophy

Year this Degree Granted: 2000

Permission is hereby granted to the University of Alberta Library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as herein before provided, neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.

W. Jonathan Baldwin
101-11716 100 Ave.
Edmonton, Alberta
Canada, T5K 2G3

Date: Aug 31/2000

University of Alberta

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Video for Low-Bandwidth Telenavigation** submitted by W. Jonathan Baldwin in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.

Dr. Anup Basu
Supervisor

Dr. Hong Zhang
Co-Supervisor

Dr. Walter Bischof

Roger W Toogood
Dr. Roger Toogood

Dr. Michael Jenkin
External Examiner

Date: Aug 30/2000

Abstract

When considering teleoperation using mobile telerobots, many factors are important. Two of these are considered in this dissertation. Firstly, a remote robot needs to be resistant to mechanical failure. This includes the video system. To provide a full environmental view, a panoramic video system using a fixed mirror provides a system not prone to failure of mechanical parts in exchange for reduced resolution images. The second factor is the amount of bandwidth required to provide real-time feedback to operators of the remote system. This feedback typically includes some form of visual data. Unfortunately, the bandwidth available for visual feedback in these applications is often insufficient.

In this dissertation, a predictive window technique is presented as a first step in reducing the bandwidth requirements and the apparent delay in visual display for the operator of telenavigation system. To achieve this prediction, a predictive Kalman filter approach with several models of mouse motion is used. This technique is applied using both panoramic and limited field-of-view images. The second step in reducing the bandwidth requirements uses feature images. These require lower bandwidth than a full video stream and can be combined with standard compression for an even greater bandwidth saving. Finally, a combination of feature images and full images is presented for use when the bandwidth limitation is removed. Experimental results are given to demonstrate that certain telenavigation tasks can still be performed successfully when using these techniques.

Contents

1 Introduction **1**
- 1.1 Problem Statement 4
- 1.2 Dissertation Outline 6

2 Related Research **8**
- 2.1 Internet Telepresence Robotics 9
- 2.2 Kalman Filters 10
- 2.3 Panoramic Video 12
- 2.4 Teleprogramming 13
- 2.5 Predictive Display 14
- 2.6 Videoconferencing and Compression 15
  - 2.6.1 Videoconferencing 16
  - 2.6.2 H.261 and H.263 Compression Standards 18
  - 2.6.3 MPEG Family of Compression Standards 19
- 2.7 Augmented Reality 20

3 Telerobot Hardware **22**
- 3.1 Telerobotic Hardware 22
- 3.2 Microcontroller Programming 27
- 3.3 Panoramic Video System 28
- 3.4 Summary 31

4 Model for Panoramic Video **33**
- 4.1 Mathematical Formulation for Panoramic Video System 33
- 4.2 Reconstruction from Panoramic Images 38
- 4.3 Panoramic stereo 41
  - 4.3.1 Theoretical Formulation for Panoramic Stereo 41
  - 4.3.2 Preliminary Results for Depth from Panoramic Stereo . 42
- 4.4 Summary 43

5 Teleoperation Interface Software **45**
- 5.1 Server Suite 45
  - 5.1.1 Shared Memory 47
  - 5.1.2 Robot Server 47
  - 5.1.3 Video Server 48

5.1.4 Communication Server 49
5.2 Client Suite 50
5.2.1 Shared Memory 51
5.2.2 Communication Client 52
5.2.3 Display Client 52
5.3 Summary 55

6 Prediction **56**
6.1 Kalman Filters 57
6.1.1 Predictive Kalman filter algorithm 58
6.1.2 Modeling Mouse Motion 59
6.1.3 Prediction Evaluation 63
6.2 Experiments 70
6.2.1 Object Location Using Limited Field-Of-View Images . 70
6.2.2 Object Location Using Panoramic Video 74
6.2.3 Object Tracking Using Panoramic Video 75
6.2.4 Measuring Image Quality 78
6.3 Chapter Summary 83

7 Low Bandwidth Feature Images **85**
7.1 Network Considerations 87
7.2 Edge Features 88
7.2.1 Sobel Operator for Edge Detection 90
7.2.2 Compression of Edge Images 92
7.2.3 Edge Lists 94
7.3 Segment Images 98
7.4 Motion Images 101
7.5 Chapter Summary 102

8 Hybrid Video for Varying Bandwidth Applications **104**
8.1 Hybrid Video 105
8.2 Combining Feature Images and Full Images 107
8.2.1 Motion Estimation and Compensation 109
8.3 Displaying Hybrid Video 115
8.4 Summary 118

9 Performance of Full, Feature, and Hybrid Video **120**
9.1 Structure of the Experiments 121
9.1.1 Navigation Experiment 122
9.1.2 Object Identification Task 123
9.2 Evaluation of Performance 126
9.3 Problems Encountered 129
9.3.1 Collisions 129
9.3.2 Robot Control 130
9.3.3 Hardware Problems 130

9.4 Summary 130

10 Conclusions and Future Work **132**
10.1 Future Work 134

Bibliography **136**

List of Figures

3.1 Mobile robot testbed. 23
3.2 Diagram of lowest plate on mobile robot. 24
3.3 Diagram of upper plate of mobile robot. 25
3.4 Tetherless video image. 26
3.5 Tethered video image. 26
3.6 Panoramic video system. 29
3.7 Full image frame from panoramic imaging system. 30
3.8 Warped image fragment from Figure 3.7 30

4.1 Single lobed mirror configuration in panoramic video system. . 34
4.2 Single lobe configuration for calculating projection from space to image plane. 37
4.3 Double lobed mirror configuration for panoramic stereo. 40

5.1 Software block diagram 46
5.2 Image request communication process. 50
5.3 X11 based operator interface. 53
5.4 GL based operator interface. 53
5.5 Numeric keypad and robot motion. 55

6.1 Mean distance error based on magnitude of the friction ratio for two different mouse motion sequences. 64
6.2 Predicted and actual values of x position in pixels for two mouse motion sequences. 65
6.3 Prediction of x position in pixels using constant velocity model. 67
6.4 Predicted and actual values of x position, in pixels. 69
6.5 User interface for object location experiment. 71
6.6 Full image from object location experiment. 71
6.7 Sample of the tracking experiment interface. 76
6.8 Summary of tracking times for seven different operators in panoramic tracking experiment. 77
6.9 Image showing blackness for mouse profile, time 18.5 seconds, when prediction is being used. 78
6.10 Image showing blackness for mouse profile, time 18.5 seconds, when no prediction is being used. 78

6.11 Percentage of blackness for each frame during a controlled trajectory. 79
6.12 First 20 seconds of x coordinate of mouse positions for two different trajectories. 81
6.13 Percentage of blackness for each frame for the first 20 seconds. 82

7.1 Masks for Sobel operator. 90
7.2 Sample edge image. 91
7.3 Small images used for edge detection and run-length encoding. 93
7.4 Subsampled edge image. 96
7.5 Line images using Hough transform. 97
7.6 Example segment image. 100
7.7 Example motion image. 101

8.1 Example hybrid image overlaying edge features on video. . . . 108
8.2 Example hybrid video frame overlaying segment features on the video frame. 109
8.3 Location of block to be matched in original frame. 111
8.4 Search region for matching block 111
8.5 Matching block in second frame 111
8.6 Overlayed images from matching process. 114
8.7 Hybrid image where the robot has rotated since the full video image was recovered 116
8.8 Hybrid video interface with motion compensation 117
8.9 Hybrid motion using double image window 117

9.1 Starting view from the robot camera. 124
9.2 Object in identification task. 125
9.3 Difficulty ratings for navigation task. 127
9.4 Difficulty ratings for object identification task. 127

List of Tables

2.1 Types of physical channels 16
2.2 Selected videoconferencing products (summarized from [34]) . 17

4.1 For Light 1, radial distance in image: $x_{10} = 27$ pixels, $x_{20} = 168$ pixels 44
4.2 For Light 2, radial distance in image: $x_{10} = 93$ pixels, $x_{20} = 200.5$ pixels (estimated). 44

6.1 Mean distance error in pixels for each predictor model. Data is from five mouse profiles based on controlled mouse trajectories. 66
6.2 Mean distance error in pixels for each predictor model. Data is from five mouse profiles based on user controlled mouse trajectories. 68
6.3 Mean distance error in pixels derived using covariance modeling. Data is from five mouse profiles based on operator's mouse trajectories. 69
6.4 Blackness calculation for first 20 seconds of 5 trajectories showing average blackness for both prediction and no prediction. . 83

7.1 Compression ratio based on resulting size and edge threshold. 92
7.2 Reduction in the size of searched Hough space vs. processing time. 95
7.3 Processing time as a function of edge image subsampling and Hough space reduction. 96

8.1 Summary of motion estimation and timing using three different image sets. 113

Chapter 1

Introduction

Historically, there has always been interest in extending workers' capabilities. Extensions can include performing tasks faster, more accurately, at a distance, and at scales not normally possible for human operators. With the growing interest in space and undersea robotics, the requirement to perform remote tasks is becoming more widespread. Besides space applications, some of the other tasks that can benefit from remote capability include explosives and other hazardous waste disposal, remote inspections and even medical applications. Indeed, any application that endangers someone performing the task, whether the danger comes from the environment or the task itself, or even any task that can be done at a distance, can benefit from the research being done with remote applications.

Teleoperation is the term used to refer to the ability to perform remote tasks using robots or other tools that extend an operator's sensing and/or manipulation capabilities. The concept of telepresence extends the ideas behind remote operation by introducing the feeling of presence [70]. The goal is to give an operator sufficient sensory feedback that they begin to believe they are physically present at the remote site performing the task directly instead of using the remote robot.

The research in teleoperation and telepresence can be categorized into many different subareas, which include:

- Telemanipulation uses teleoperation/telepresence techniques for remote manipulation tasks [41, 73]. This research area includes issues concerning control of manipulators, appropriate sensor feedback for manipulation tasks, and scaling the actions of the operator to the range of the manipulator (from micro-surgical manipulators to large industrial robot arms).

- Telenavigation is concerned with navigating remote mobile robots. This research area can be coupled with the telemanipulation area for such applications as hazardous waste disposal [41] and space applications [51].
- Telemedicine is telepresence for medical applications [1, 37]. These applications range from diagnostic tasks to surgical tasks to physical therapy and home care.
- Telemanufacturing uses telepresence techniques for manufacturing tasks. This can be related to telemanipulation, but also includes issues concerning distributed manufacturing (with more than one site involved in the construction of a given product) and shared control of the manufacturing task from more than one site.

There are several general research areas that are applicable to most of the areas of telepresence described above. Control theory for teleautonomous systems [31] researches issues in control modes and how they apply to telepresence applications. These control modes range from teleoperative control where the user directs every action of the remote system to supervisory control where the operator oversees the actions of an autonomous system. The control modes can be combined to allow greater flexibility for the remote system as well as allowing multiple operators to be involved in the task being performed. Another issue in control theory is compensation for delayed feedback. Feedback delay can cause complications in the task being performed if the control mode is such that the remote system cannot react to conditions without directives from the operator.

Another of the general research areas is concerned with bandwidth restrictions on telepresence applications. There can be many reasons why bandwidth is a problem. Low bandwidth communications links, transmission delays, and a lack of processing power all contribute to bandwidth limitations. Other issues involved in communication for teleoperation applications include signal dropout and latency. This research is interested in how image data can be displayed when there are bandwidth limitations.

A telepresence system consists of a human operator with an interface to a remote robot. The remote robot usually includes at least one camera. This camera generates a significant amount of video data every second. Consider a simple 512x512 8-bit grey scale video sequence. Each frame requires 256 Kbytes of data. With a frame rate of 30 frames per second, approximately 8 Mbytes of data is transmitted every second when no compression is used. Even using standard compression techniques, this bandwidth requirement cannot be reduced enough to use a traditional phone line (28.8 Kbits/second). For this, each frame needs to be reduced to approximately 124 bytes. This requires more than 2000 fold compression.

Increasing the amount of bandwidth available is one way of improving the situation, but unlikely to solve the problem completely in the near future. Changing the way visual information is displayed offers an alternative when only limited bandwidth is available. Reducing the frame rate would increase the amount of data allowed per frame, but even reducing the rate to 10 frames per second would only allow approximately 400 bytes per frame.

To allow the operator a view of the environment, there are several hardware approaches that can be used. The simplest is to mount a fixed, limited field-of-view camera on the mobile robot. In order to see the full environment, the operator must move the robot around. This is time and energy consuming, but the operator only sees what they want to see.

An alternative to a fixed camera is to mount the camera on a pan-tilt unit. The operator is then provided control to this unit and can move the view point around in the environment with a few limitations. Use of a pan-tilt based video system allows a larger view of the environment than a fixed camera without moving the robot. If the robot is located in a dangerous position, moving it can be undesirable.

There are limitations inherent in traditional pan-tilt units. The first is that the range of motion is usually constrained for mechanical and electrical reasons. Limited range motors provide a mechanical limitation to motion while cables also keep the unit from unconstrained movement. There is also a time constraint involved. There is a delay from when the operator requests a change of view and when the change occurs. As a benefit, a camera mounted on a pan-tilt unit can have the same resolution as a fixed camera.

The third alternative for allowing the operator a full view of the environment is to use a panoramic or omni-directional video system. This system allows a full 360 degree field-of-view without any mechanical constraints. A panoramic system can give the full view to the operator all the time, or a subsection can be displayed. Changing this view requires no mechanical actions and can be done almost immediately, barring transmission delay if there are network constraints.

The major drawback to a panoramic system is the low resolution compared to either of the other two methods described above. Most fixed panoramic systems use the same types of cameras as pan-tilt units. Since the panoramic system stores more information in the same video image than a limited field-of-view image, the resolution for specific portions of the image is lower than for pan-tilt. The trade-off is resolution for speed and no mechanical parts to break.

1.1 Problem Statement

In traditional teleoperation applications, a remote site is usually connected with the master site using a dedicated high-bandwidth communication link. For these applications, bandwidth is not a problem and using conventional compression or bandwidth limiting techniques produces a good video stream. When only limited bandwidth is available, the video quality will degrade. This results in a degradation of the image quality or a reduced number of frames per second or a combination of the two.

If the remote telerobot is located far away, possibly on another continent, using a dedicated communication line is not feasible unless the teleoperation system is supported by a government or other agency with a large expense account. The most common communication link is the Internet. The Internet connects almost every part of the globe, but can have significant bandwidth restrictions. With the number of people using the Internet on a daily basis and the wide variety of network media used, the average apparent bandwidth is no better than a traditional telephone line.

With a distant telerobot, a robust system is required. If mechanical problems occur, the operator will be unable to fix the problems and it might take time to contact a telerobot maintainer and get the problem fixed. Using a panoramic video system allows the operator to view the full environment with minimal chance of mechanical failure of the video system (the robot itself may or may not be robust, independent of the video system).

For acceptable teleoperation over low bandwidth communication links, the amount of data needs to be reduced from what would be transmitted over a higher bandwidth link. Because most teleoperation involves visual data, and since this takes up a significant percentage of the available bandwidth, reducing this data is the first step to reducing the amount of bandwidth required.

Current research on low bandwidth teleoperation reduces the amount of visual data by either replacing it with graphics generated at the operator site based on information extracted at the remote site, the case for teleprogramming based teleoperation, or by transmitting images only after an action has been completed, as used in World Wide Web based teleoperation. Extracted feature information is much more compact than full video, but some of the important information can be lost depending on the extraction method.

Techniques used to create feature based images for low bandwidth teleoperation can also be used for higher bandwidth teleoperation. For the higher bandwidth applications, the video stream can be enhanced by overlaying low bandwidth, feature based video on the higher bandwidth, reduced frame-rate, full image sequence. This will produce a hybrid video stream.

Another approach is to transmit only a limited subimage of the full camera image available. Using this as a starting point, the transmitted image size can be reduced further by using compression and other techniques like those mentioned above. One way to determine the size of the subimage is to use a predictive window. A predictive window would determine the section that the operator is viewing currently and, based on past activities, what they will want to view in the next image update.

The problem studied in this research can be summarized by the following statements:

Are there alternative methods for displaying visual information in low-bandwidth telepresence applications using panoramic video systems which are effective? Can these methods be used to enhance video streams when there is more available bandwidth?

There are several issues that will be explored. These include:

- Can a predictive display be developed that will allow an operator to view a small section of the camera image and still be able to operate a remote robot? Can this predictive display adequately predict operator viewing actions?
- Can usable imagery be produced by displaying interesting features extracted using various image processing techniques? These features can include edges, segments, and differences between successive frames.

1.2 Dissertation Outline

This dissertation presents research performed in teleoperation using a mobile robot equipped with a panoramic video system. This teleoperation is performed over the low bandwidth, on average, available on the Internet. The panoramic video system is used to add a robust video system while still allowing the operator to have access to a full view of the remote environment with minimal robot motion.

The proposed solution to the previous mentioned problem is in two parts. The first uses predictive windows to reduce the size of the images that are being transmitted. This uses a predictive Kalman filter to predict user mouse input which determines a viewing direction in the panoramic video stream.

The second portion of the solution first generates and transmits edge feature images, after investigating both segment features and difference images. When the bandwidth available is large enough that transmitting feature images does not require the full communication capabilities, the feature images are combined with a low frame rate video stream to provide the operator with a hybrid video display of both interesting features and the full context found in the regular video stream.

Chapter 2 presents previous research that is related to the dissertation topic. This research includes: Internet telerobotics, an outline of applications using Kalman filters, panoramic video applications and systems, teleprogramming, predictive displays, videoconferencing and compression, and augmented reality.

A description of the complete platform used in the research is provided in Chapters 3, 4, and 5. Chapter 3 describes the telerobot hardware, while Chapter 4 presents the mathematical formulation used to interpret the panoramic video stream. Chapter 5 completes the platform discussion with an overview of the interface software written to allow teleoperation.

The experimental description and results found in Chapter 6 discuss the predictive window approach to reducing bandwidth. Chapter 7 discusses feature images and their use in this application domain. A description of hybrid video concepts is found in Chapter 8. Chapter 9 describes the experiments conducted to evaluate the performance of feature images and hybrid video. Finally, Chapter 10 presents some conclusions derived from this work and possible directions for future work.

Chapter 2

Related Research

This dissertation presents a set of techniques that will allow teleoperation of a mobile robot using panoramic video over networks, such as the Internet, with a range in available bandwidth where the lowest end is equivalent to that provided using telephone lines. These techniques cover several different research areas. The major areas involve Internet robotics, a fairly new field, Kalman filters, and low bandwidth teleoperation.

The issues of teleoperating a robot over low bandwidth communications links have been explored using a number of different techniques. These include teleprogramming or model-based telerobotics and predictive displays.

Two additional related fields are video compression and augmented reality. Video compression can be used in addition to this work to reduce the required bandwidth for a given video stream. Augmented reality studies applications where actual video streams are modified by graphics to create a combined real and synthetic world.

This chapter is organized as follows. First, Section 2.1 will discuss some of the research being done with Internet based robotics. The following section, Section 2.2, will describe selected research in the very large set of applications involving Kalman filters. Panoramic video will be discussed in Section 2.3. Sections 2.4 and 2.5 will discuss teleprogramming and predictive displays. Finally, Section 2.6 will give a brief overview of some of the more common video compression techniques, while Section 2.7 will discuss research in augmented reality.

2.1 Internet Telepresence Robotics

Internet telerobotics has become a popular research area in telerobotics. In 1994, two telerobots were connected to the World Wide Web. One in Australia [74, 75] at the University of Western Australia and one at the University of Southern California [29]. Since that time numerous "robots" have been connected to the Internet, some specifically to the Web, including a coffee pot, several pop machines, and even a garden.

The telerobot in Australia [74, 75] has been online since 1994, although its hardware and software have both been updated in the years since. This robot currently consists of a robot arm that can stack and unstack blocks in a restricted workspace. The operator interface is a set of web forms that allow the operator to specify commands to control the arm. A single image frame is transmitted when a command is received and processed by the robot server. This process takes several seconds per command. One of the software updates now allows users to issue complex commands from the web form. This allows experienced users to issue a single complex command and be informed when it is completed instead of having to wait for each simple command to be completed before entering the next one. The average time between requests is about 55 seconds.

The USC telerobot [29] called the "Mercury Project" was taken off the web in 1995 and replaced with a telegarden. During its time on the Internet, it had been accessed by over 50,000 sites around the world. The Mercury Project consisted of a robot arm equipped with a single camera and a pneumatic system to blow compressed air. The arm was placed in a semi-annular area that was filled with sand and buried artifacts to simulate an archaeological dig. The arm was able to move around in a semi-circular arc over the sandbox, raise and lower the end of the arm, and blow compressed air at the sand. The goal for the operator was to uncover the buried artifacts and determine their relationship to each other (the solution can still be found on the USC web page). The operator interface consisted of a point and click interface for moving the robot around and blowing air at the sand. A current status image is presented after each command to the robot once the processing is complete. The response time for this interface is between 10 and 60 seconds depending on the location of the operator and the available bandwidth.

One of the most severe restrictions on using a web based teleoperation interface is the update rate. The update rate is affected by both the bandwidth limitations inherent to the Internet as well as the overhead incurred by using the World Wide Web. In a navigation task, being able to see where the robot is at a rate similar to video frame rate is almost a necessity. Most of the web based interfaces have an update rate of several seconds per frame (some of the fastest web cams, without attached telerobots, can update about every 2 seconds). For a navigation task, and ideally for any teleoperation task involving interaction with the environment, we require a video update rate between 5 and 30 frames per second.

2.2 Kalman Filters

Kalman filters have been used for a very wide variety of applications. These applications include navigation in space, tracking objects using radar data, and robot localization using data from multiple data among a great many others. Generally, Kalman filters can be used to derive an estimate of the state of a system based on a linear function of the all previous states and current measurements which minimizes the error variance. Of interest to this dissertation are those applications that relate to robotics and prediction.

Kalman filters have been shown to be an effective method of prediction in a number of applications. Emura and Tachi [22, 23] are tracking head motion for head mounted display technology. They propose a multiple sensor system that tracks both orientation and angular velocity of an operator's head. They use a Polhemus Tracker as a base sensor and gyro sensors to compensate for the latency and low sampling rate of the Polhemus. They use a predictive Kalman filter to combine the data from both sensors and to predict head motion.

Green *et al.* also use a Polhemus Isotrack, in [46], to track head motion and a predictive Kalman filter to predict future head motion. The authors assume that the head motion can be modeled as a Gauss–Markov process. They use the filter to determine predicted head orientation and position for use in calculating line-of-sight and viewpoint in a virtual reality application.

In [42], a Kalman filter is also used to predict head motion. In this paper, the authors model motion using a simple kinematic model assuming constant acceleration for their filter. This model is used in a visually coupled control system activated by head motion. This paper also compares the use of the predictive Kalman filter with a polynomial based prediction method and shows that the Kalman approach produces better results for both smooth and abrupt head motion.

Kalman filters have been used in many other tracking applications. In addition to estimating or predicting current input device states, these filters can be used when tracking objects, usually in video streams. One use in these types of applications is in estimating the current position of an object being tracked [53]. Kalman filters can also be used to reduce the effects of visual latency in active tracking applications [14].

In [39], Jaynes and Hou present a Kalman filter based algorithm to register synthetic and real video images in their augmented reality application to maintain current pose information relative to a head-mounted camera system. This system maintains temporal registration of objects and their relevant graphical models in a manufacturing application.

Other applications that use Kalman filters include mobile robot navigation [13, 18, 20, 52, 66] and localization [25, 40, 60]. Most of these problems involve both sensor fusion and position and/or attitude estimation. Robots can mount several types of sensors in order to improve the accuracy, and redundancy, of their environment sensing. The data from these sensors needs to be compiled and an estimate of the inaccuracy derived. A Kalman filter approach to this fusion is a common solution to the problem [24, 25, 60].

2.3 Panoramic Video

Panoramic images can be obtained from several sources. First, these types of images, with a 360 degree view, can be obtained using multiple cameras. When using multiple camera views, the resulting images must be stitched together to generate a synthetic image panorama. This approach to generating 360 degree environment views is limited by available processing, to do the stitching, the cost of multiple cameras, along with the problems related to registering the different images. The resulting images can have similar resolution to standard NTSC images.

A second method of generating panoramic images involves a single camera that swivels around a central axis. This type of camera can take multiple images or a single central strip, as in [45], and then stitch them into a single panoramic image similar to the method above. Another single camera method uses a high resolution scanning camera such as the one produced by TelePhotogenics Inc. [38]. This camera can generate images with a resolution of approximately 4000 pixels by 40000 pixels over 360 degrees. The major drawbacks to this approach are the time required to take a single frame and the memory requirements to process the image.

The most common method for generating panoramic images today involves a single camera and a mirror configuration. This allows a single standard size image to contain the full environment view. The trade-off is that the resolution is reduced significantly to allow for the full panoramic image. These mirrors can be conic [17, 56], parabolic [11, 28], or hemispherical [5].

These panoramic, or omni-directional, sensors are used in a variety of applications. The authors of [45] present a method, using a swiveling camera, to accurately determine camera motion. In [76], the authors present a technique which fuses information from both omni-directional vision and ultrasonic sensors. This method is used to generate a map of the local environment. Boult *et al.* [10] present a multi-body tracking system using an omni-directional video system. This system is being evaluated for its application to surveillance.

Panoramic video systems can also be used to generate stereo images. In

[28], the authors use two panoramic camera systems mounted vertically to generate their panoramic stereo images. The advantages to this method are that each image contains a full panorama and both images can be captured at a single instant.

A second arrangement for generating panoramic stereo images is presented in [21]. In this paper, the authors use a single panoramic sensor [17] which is mounted on a rail to allow the sensor to make known translations between two points and generates an image at both ends. While having two full panoramic images allows for reasonable quality images, the need to take two images at different locations with the same camera requires more time for image acquisition. This system is used in mobile robot localization algorithm which is shown to be robust.

2.4 Teleprogramming

The concept of teleprogramming was introduced as a method of allowing real-time operator control of a telerobot in the presence of communication delays [27]. This method of control, also called model-based telerobotics, uses graphics technology to simulate the remote environment and allows the operator to interact with this simulation. The simulation can take the form of virtual reality or a more simple graphical display. Funda *et al.*, in [27], use the teleprogramming control methodology to allow an operator to experience kinesthetic feedback as well as visual feedback in real time, while the remote arm executes the specified actions after a delay. When an error occurs at the remote site, sufficient information is returned to the master site to enable the teleprogramming environment to return to a state just before the error occurred so that the operator can determine what went wrong and how to solve the problem.

Teleprogramming has also been used in sub-sea teleoperative research [65]. In this research, the authors chose to include visual imagery in the teleprogramming environment. The real images are, by necessity, of low quality and are not received by the master site for several seconds (the delay is assumed to be about 10 seconds). Because of the bandwidth limitations, not all of the image data can be transmitted to the operator. To determine what information is desired, the authors have designed an intelligent image fragmentation method, also described in [64]. This method uses predefined "characterizing points" to determine which image fragment would be the most useful to the operator at the current time. They use a weighting for each point based on the current action and which image fragment contains the most useful set of points.

Another application of model-based telerobotics can be found in [47]. In this paper, the authors describe their system which is used to investigate tasks requiring manipulation of known objects. One of the central problems in teleprogramming applications is the acquisition of the world model used at the operator site. The system presented in the paper uses a fast vision system to recognize objects and transmit a spatial location to the operator site. The operator's interface consists of the standard graphical interpretation of the remote site as well as slower, low frame-rate video of the actual remote site.

2.5 Predictive Display

A predictive display attempts to compensate for communication delay or bandwidth limitations by predicting either the operator's actions or the actions of the remote system. Predicting the operator's actions will allow the system to request more information from the remote site which can be displayed while waiting for the next update. A prediction of the actions of the remote system allows a model of the predicted actions to be displayed before the actual events occur.

There are two types of predictive displays that can be used when low bandwidth or large communication delays are present. The first type of predictive display uses a combination of real video and graphics (similar to augmented reality discussed below in Section 2.7). Bejczy *et al.* [8, 7] describe time-delay experiments done using a predictive display to simulate space applications. Their system uses graphical overlays, of varying transparency, with real-time video. The graphics are used so that the operator can preview and predict the results of their commands. This allows the operator to generate a continuous sequence of commands for the robot to complete an action without the stop-and-wait associated with large communication delays.

A second type of predictive display used for low bandwidth communication links was studied here at the University of Alberta [3, 4, 5]. This research is examined in Chapter 6 along with further experimentation. This type of display reduces the amount of data transmitted over the link by predicting what portion of a large image the operator is going to be viewing during the next time period. This type of display can be combined with standard compression techniques to reduce the bandwidth required still further. The operator's next viewing portion is predicted from their current viewing position and previous positions using a predictive Kalman filter. This Kalman filter uses a physical model of the mouse, the input device determining the current viewing position, and can be modified to use a different model consistent with another input device.

2.6 Videoconferencing and Compression

When trying to reduce the bandwidth of an image stream, the traditional first approach is to use a video compression algorithm. There are two major standards for video compression, H.26x and MPEG, that can be used. There are also other, non-standardized, compression techniques published, including spatial varying resolution techniques [6, 77].

In this research, teleoperation over the Internet is being examined. The Internet is made up of many different networks and network technologies. All of these networks have their own capacities and users. Since most of the networks are multi-access, the bandwidth available to any given user is small portion of the theoretical maximum for the network. Some of the standard physical channels used in networks and their capacities are summarized in Table 2.1 (summarized from [9]). We can see from the table that the higher bandwidth networks are multi-access which means that the maximum available bandwidth is divided among several users. In some cases, the available

Channel	Bandwidth	Multi-access
analog telephone line	theoretical max 25,000bps	No
ISDN	2 channels of 64 Kbps plus 1 channel of 16 Kbps	No
T1	1.544Mbps	Yes
Ethernet	10Mbps	Yes
T3	44.736 Mbps	Yes
FDDI	100Mbps	Yes
Broadband ISDN	155 Mbps	Yes

Table 2.1: Types of physical channels

bandwidth is less than the maximum because of congestion problems in the network itself. The two networks that are not multi-access, telephone lines and ISDN, both provide only low bandwidth and even then, ISDN is still not commonly available to the average user.

We have see in Section 2.1 that both access times and response times for a single operation on an internet robot are long. In some cases, these times can be as long as one or two minutes. This is despite the fact that the Internet is made up of networks with bandwidths on the order of megabits per second. From personal use, an average bandwidth similar to the theoretical maximum for analog telephone lines, about 25 Kbps is expected.

2.6.1 Videoconferencing

Videoconferencing (or teleconferencing) is another area related to telepresence. In a videoconferencing application, two or more people are able to talk to each other, as in a traditional telephone conference, and also view each other by transmitting live video image streams over a network. The issues of how and what to transmit are similar to those in teleoperation. Determining what are important features in an image and how to use this information for compression purposes are two issues that appear in both teleoperation and videoconferencing research. One difference between the two areas is the types of features that are considered important. In videoconferencing, faces and other personal features are generally considered important, while in teleoperation, these are not important features.

Product and Provider	Network Technology	Video Compression Standard
Audiovision (Smith Micro Software)	28.8K Modem	H.261
CU-SeeMe v 3.1 (White Pine)	28.8K Modem	H.263
DECspin (DEC)	ISDN	JPEG
InPerson (Silicon Graphics)	ISDN	H.261
Internet Video Phone (Intel)	Internet	H.263
NetMeeting (Microsoft)	28.8K Modem	H.263
ShowMe (Sun Micro Systems)	Internet	CellB

Table 2.2: Selected videoconferencing products (summarized from [34])

Videoconferencing technology can be used for more than just a pair of people meeting. It can be used for teaching courses, holding world-wide meetings, remotely viewing conferences or seminars, or any other application that has one person seeing and hearing what another person is doing. There is research being done on telelearning, for example, which applies the same sorts of videoconferencing techniques to education.

There are many commercial videoconferencing products available. They are available for most computer platforms including PCs, Macintoshs and most versions of Unix. From a brief sampling of the available products (selected products are shown in Table 2.2 summarized from [34]), we can see that while there are some products that use standard telephone lines, most of them use ISDN technology or communication links that provide even higher bandwidth. For the products that use telephone lines, H.263 video compression seems to be the most popular standard (see Section 2.6.2 for more details).

While there are videoconferencing products available that can use telephone lines, they are based on video compression standards that were designed for ISDN. The available bandwidth for ISDN is at least 64 Kbps. Due to the combination of requiring lower bandwidths than most videoconferencing packages support and different features of interest, using available videoconferencing products is not appropriate for very low bandwidth telepresence applications.

2.6.2 H.261 and H.263 Compression Standards

As previously shown, H.261 and H.263 are the video compression standards designed for videoconferencing. H.261 was designed by the ITU (International Telecom Union), to be used for video transmission over multiple channel ISDN connections. Using it over phone lines with less than half the bandwidth can significantly reduce the performance.

One of the means by which the H.261 standard reduces the bandwidth required is by using small image sizes. It also includes parameters to reduce the frame rate of the video stream. A video stream constructed of images in the Quarter Common Intermediate Format (QCIF) can be compressed using the H.261 standard to be viewed at either 10 or 30 frames per second. In a QCIF image stream, each image contains 144 lines with 176 pixels per line. This is approximately one twelfth the size of a standard NTSC video stream.

The H.263 standard was designed to allow video compression at rates lower than 64 Kbps [78]. A good review of this standard can be found in [68]. This standard, while it was designed for the very low bandwidth videoconferencing, has several problem areas in relation to this dissertation.

The first problem with the H.263 standard is that the video images are assumed to contain a talking head. This means that the content usually consists of head of the speaker, possibly the shoulders too, and a static background. There is very limited motion in the video stream. The resulting spatial and temporal redundancies allow for a large reduction in the resulting bitrate.

The second problem is the size of the image stream. Similar to its predecessor standard, H.261, the H.263 standard achieves some of its compression by using very small images. In addition to supporting QCIF images, the standard also allows SQCIF which is about one half the size of a QCIF image. This standard allows optional support of larger image formats, but the bitrate will not be reduced to the same levels.

2.6.3 MPEG Family of Compression Standards

The second major family of compression standards is that produced by the Moving Picture Experts Group, MPEG. This set of standards currently has three finalized standards, MPEG-1, MPEG-2, and MPEG-4, along with several under development. The first two standards provide high bandwidth video transmission while MPEG-4 supports a much wider bandwidth range.

MPEG-1 was the first of the MPEG standards developed. It allows for bitrates of approximately 1.5 Mbps. This standard is used generally for accessing audio and video streams from storage media such as CDs [32]. While it is possible to have bitrates as low as 64Kbps using MPEG-1, significant reduction in the quality of the video stream occurs.

Following the release of the MPEG-1 standard, a standard for much higher bandwidth was developed. This standard called MPEG-2 is intended for rates of five to 10 MBps. Applications such as DVD and digital television were the intended target.

The most recent finalized standard in this family is MPEG-4 [43]. This is a broadly based standard to allow for compression of a large variety of video and audio information. This includes both natural and synthetic images and aural objects alone and in combination. The video stream is composed of individual media objects consisting of descriptive elements. These objects can be represented independently of the surroundings.

The MPEG-4 standard allows for compression bitrates as low as 5 Kbps under the Very Low Bit-rate Video (VLBV) Core. The VLBV Core provides tools and techniques to allow video compression on low resolution image streams such as Common Intermediate Format (CIF) which is about 4 times greater than QCIF described above.

All of the video compression standards described in this section have one major drawback for this research, with the possible exception of MPEG-4: they all use low resolution images and these images have a constant size. The prediction process described in Chapter 6 can generate streams containing images of varying size. Not only are these images larger than those generally supported by the standards, they can change from one frame to the next. MPEG-4 does provide support for varying size image streams, but it is unclear if they can be supported when the available bandwidth is quite small.

2.7 Augmented Reality

Augmented reality is the combination of virtual reality and telepresence, overlaying graphics based on either real world knowledge or completely virtual information. Some of the predictive displays use this technique to allow operators to experience real-time simulation of their commands while waiting for the robot to perform the requested actions. The techniques of augmented reality have been used in both the teleprogramming and predictive display areas. A recent survey of the research being done in augmented reality can be found in [2].

Augmented reality has been used in a number of applications. In [19], the authors discuss a system to create and verify graphical models of remote environments using a combination of virtual reality and telepresence technologies. The telerobotic applications that were being studied at Sandia National Laboratories usually required the construction of a graphical model based on remote data. This model is used by the remote robot to generate its own motion plans. In order to facilitate this model construction, the authors proposed a system that would allow the human operator to interactively identify objects to the vision system. The vision system would then extract the depth and position information for the model. The same system can be used to verify this model by displaying the objects in wire-frame form overlayed on the real-time video based on the current position and orientation of the remote robot.

Another application where augmented reality can make a significant improvement on performance is situations where the video data is degraded due to the environment, such as under water, in smoky areas, etc. The authors of [48] study visual perception aids that can be used in such situations. The described system uses known model information to generate a graphical model that can be animated in real-time to overlay on real environment data obtained from video cameras. The graphical model is updated by the operator by selecting important objects from the real scene. The full animated model is coupled with the video data and presented to the operator as a single view that is updated as the task progresses.

The authors of [54] propose a virtual reality system that can be used for teleoperation in environments with poor visibility. In their system, they use a predefined model of the environment that needs to be calibrated to compensate for unknown parameters, such as errors in robot position and orientation. This calibration requires the participation of a human operator to match up control points in the graphical model and the corresponding points in the real images. Once this calibration has been completed, the operator works in the virtual reality environment. It is possible to use the same system to operate in an augmented reality environment with wire-frame models overlaying the real camera images.

Chapter 3

Telerobot Hardware

There are many possible types of telerobots that can be used in teleoperation research. Many of the internet based robots are either telerobotic arms or hands. I am interested in mobile robotics and telerobotics and chose to use a mobile platform for this research.

Mobile robots are becoming widely available, although not inexpensive. To convert these mobile robots into telerobots, a video system will usually need to be installed. This is not a difficult task, but a necessary one. At the start of this research program, a mobile robot was constructed rather than purchased.

This chapter will discuss the telerobot hardware used in this research into video for low-bandwidth teleoperation. Section 3.1 will discuss the robot platform itself. The programming environment of the microcontroller will be discussed in Section 3.2. Finally, in Section 3.3, the panoramic video system will be discussed.

3.1 Telerobotic Hardware

The implementation of the research presented in this dissertation is designed to be applied to a navigation task in an indoor environment. In order to conduct the experiments, a mobile telerobotic platform, shown in Figure 3.1 was designed and constructed in the Department for this research project.

The mobile platform was designed by Dr. Ron Kube, a former PhD graduate of the Department, based on a design he used in the past. It consists of a set of aluminum plates, one foot in diameter, connected by threaded rods

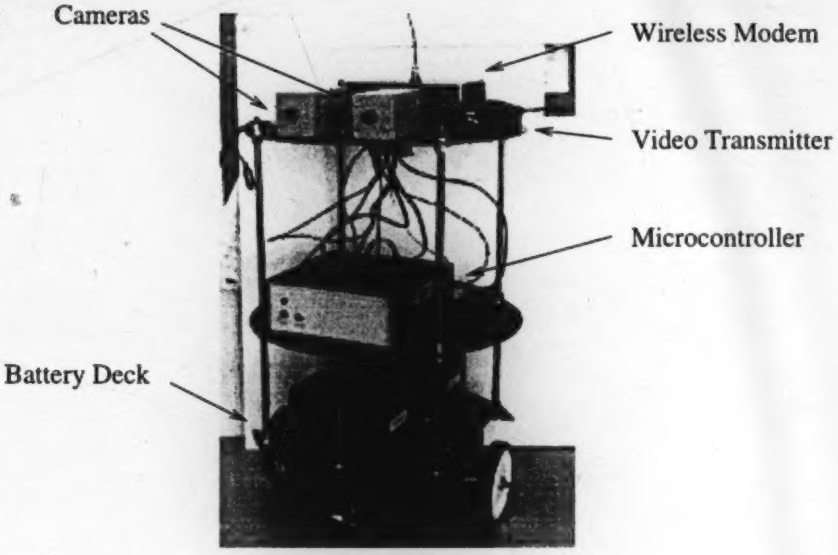


Figure 3.1: Mobile robot testbed.

to create a robotic platform that is approximately two feet tall. The plate heights are adjustable to accommodate any equipment or sensors that the robot is capable of carrying.

Since the robot has movable plates and components, only the default configuration of the components will be described. The only components that are usually moved from the default location are those of the video system. Moving these parts allows the point of view of the video stream to change as well as changing the video stream between limited field-of-view and panoramic view.

The lowest of the three plates mounts components on both the upper and lower surface. The lower surface of the plate mounts the two motors which provide two degree-of-freedom motion. The motors are on opposite sides of the plate and driven independently. Two castor wheels with adjustable height are also mounted through the lower plate. These wheels provide stability to the robot platform. Figure 3.2 provides a diagram showing how the components are attached to the underside of the lowest plate.

The upper surface of the lowest plate mounts two 12 volt rechargeable batteries. One battery provides power only for the motors. This allows the motors to draw more current without overloading the more sensitive components (the

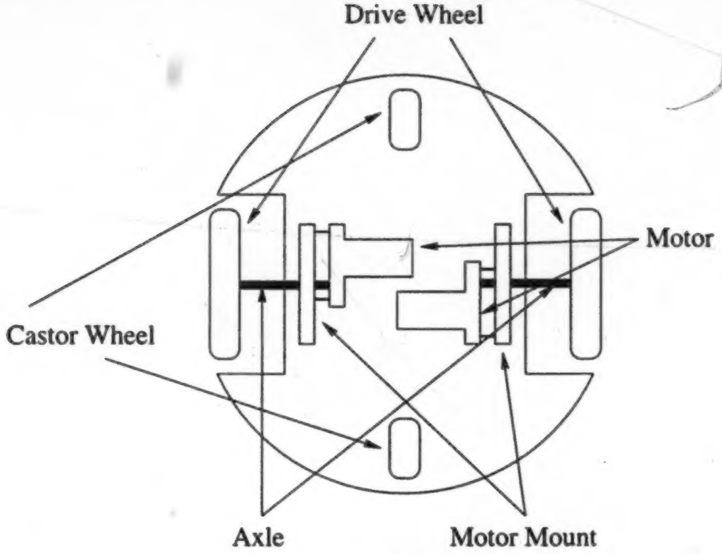


Figure 3.2: Diagram of the underside of the lowest plate on the mobile robot. The motors are placed on opposite sides of the plate with castor wheels located between them also on the circumference.

microcontroller for example). The second battery supplies power for the rest of the components including the microcontroller itself, the video system and the wireless modem. Each of these batteries can be removed and recharged independently.

The middle plate contains the microcontroller and supporting electronic modules such as the motor control and DC power regulator. The microcontroller is a Motorola 68HC11 microprocessor. The microcontroller was purchased from New Micros Inc. in their NMI-0021 single board computer. The NMI is capable of mounting up to 64 KB of RAM, but this application only requires the basic 8 KB. The microcontroller provides motor control and communicates to a base station so the operator can direct the robot.

To provide communication with a base station, the robot is equipped with a wireless modem, usually mounted on the upper-most plate, while the base station has a second wireless modem. Each of these modems is constructed using a ProxLink Radio Module from Proxim Inc. The modem connection allows wireless serial communication between the base station and the robot

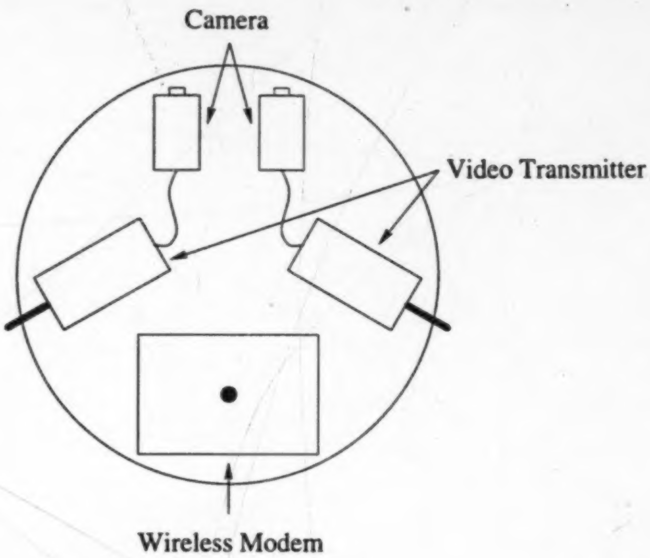


Figure 3.3: Diagram showing the layout of the upper plate with tetherless limited field-of-view video system.

at a data rate of up to 19.2 KBaud with an indoor range of up to 500 feet [58].

The upper-most plate is also the standard location for the video system. This plate also has a small power distribution module mounted on the underside to allow the different logic components to be powered from a single battery and also provide a power switch to disconnect the power for all these components at the same time. Figure 3.3 shows a diagram of how the components are mounted on the upper plate when the full tetherless limited field-of-view video system is mounted.

The original limited field-of-view video system consists of a pair of colour cameras installed on the top plate of the robot as can be seen in Figure 3.3. These cameras provide a view from the front of the robot that can be used to provide stereo vision when the cameras are aligned properly. The cameras are each connected to a video transmitter which is paired with a receiver that can be connected to a standard NTSC video input port of a computer workstation. This allows a wireless video feed for tetherless operation. Figure 3.4 shows a sample image taken using this video system configuration. Of course, the cameras can be connected directly to the NTSC port to generate better video

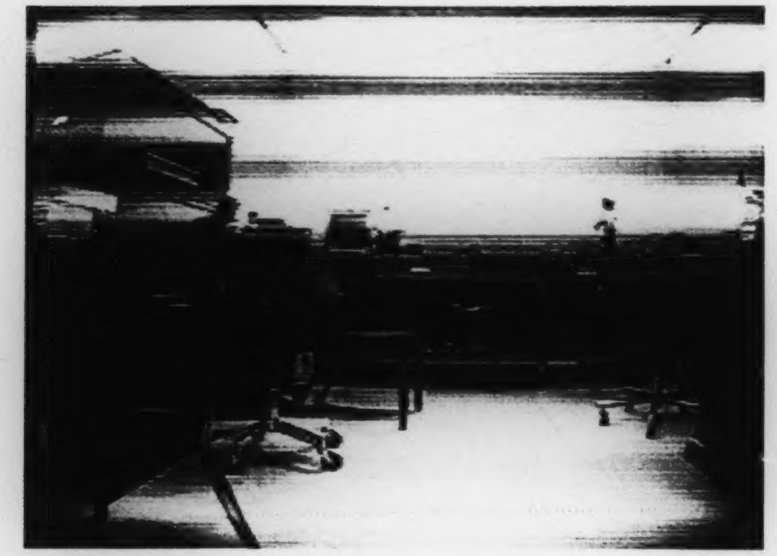


Figure 3.4: Image taken from a single camera in the tetherless limited field-of-view video system.

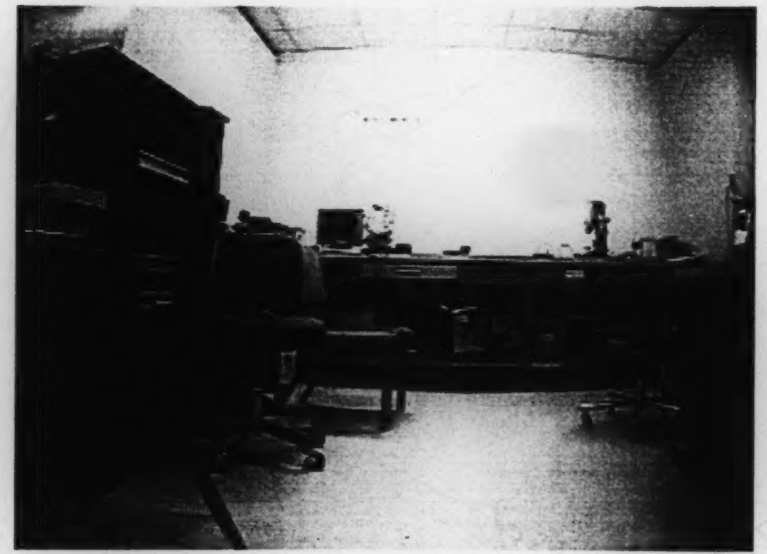


Figure 3.5: The same scene as Figure 3.4 taken from a single camera in the limited field-of-view video system when wired directly to the video input port.

images. With the wireless video, there is a large amount of interference in the image which is due partly to the environment where we are using the robot as well as other factors such as the transmission process itself. Compare the image in Figure 3.4, which shows the quality of the image when the telerobot is used in a completely tetherless application, with the image in Figure 3.5 which is wired directly to the input port.

3.2 Microcontroller Programming

The HC11 microcontroller is programmable using the Forth language. This version uses MaxForth supplied by New Micros Inc. Forth is an interpreted language meaning there is no compilation step, simply defining words and having the environment interpret them. It is possible to get different programming environments, including a C compiler, but using Forth for this application is straight forward and provides the necessary capabilities.

Using the Forth environment allows for simple communication between the robot and the base station and also simple words to send signals to the motor control circuitry. The basic microcontroller environment provides a word prompt where the user types in various word definitions and can have them interpreted immediately. Using a serial connection, the robot is equipped with a wireless modem, the MaxForth environment can be accessed directly and all of the words are available.

The microcontroller at this time only controls the motors and their direction. Since there are no sensors except the camera systems, there is nothing else for the microcontroller to manage. The amount of memory available to the controller, 8 KB of RAM which can be expanded to 64 KB, is too small to perform any reasonable image processing. Using a different on-board controller, a modern laptop for example, would allow for on-board image processing, but this is not necessary at this time.

The motor control consists of only those words necessary to change the direction and speed of the two motors. These words include:

FORWARD This engages both motors forward.

BACKWARD This engages both motors in reverse.

STOP This word completely stops both motors no matter their current state.

LEFT-ROTATE This will engage the left motor in reverse and the right motor forward. LEFT-ROTATE provides a rotation about the centre of the robot to the left.

RIGHT-ROTATE This will engage the left motor forward and the right motor in reverse providing a right rotation about the centre of the robot.

LEFT-TURN This word stops the left motor while engaging the right motor forward. This provides a turn towards the left pivoting about the left wheel.

RIGHT-TURN This stops the right motor and engages the left forward pivoting the robot about its right wheel.

While these Forth words only provide a minimum amount of motor control, they are enough to allow an operator to move the robot around in the environment. There is no feedback from the motors or additional sensors to allow the robot to accurately control its own motors, so other words are unnecessary. Motor feedback isn't strictly necessary since a human operator is involved in the control loop. Adding words to allow a change in speeds is easily accomplished using the Forth environment and using this capability with the existing words is also accomplished with ease. If additional capability is required from the robot, such as additional sensors or intelligent behaviours, adding more words to the Forth dictionary can be accomplished quickly and can incorporate the words that already exist.

3.3 Panoramic Video System

In nearly all teleoperation applications, the operator wants to view the environment. The ability to view the whole environment without moving the telerobot can make this action more efficient in both time and energy expended. As discussed in the Introduction, two possible approaches to viewing

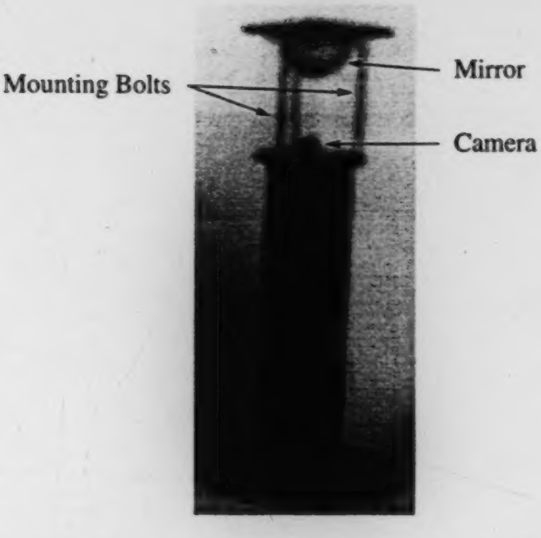


Figure 3.6: Panoramic video system unmounted from the telerobot. The mirror, camera, and mounting bolts are indicated.

the full environment are using a traditional pan-tilt unit or a panoramic video system. There are benefits and drawbacks to both systems. In this dissertation, a panoramic video system is used to allow the operator to view the full environment.

The mobile platform was originally designed for use with a pair of NTSC cameras to provide limited field-of-view stereo vision. It is also able to have the panoramic video system mounted. This panoramic system, shown in Figure 3.6, consists of a camera which has been fitted with a hemispherical mirror to provide panoramic video. The camera system provides standard NTSC video, but the images contain panoramic information. The image in Figure 3.7 – a panoramic view of our laboratory – shows what these images look like before they have been processed for viewing by the operator. The outer rings in the image are part of the mounting system for the mirror. The next inner ring is the actual panoramic information, while the inner-most rings are reflections of the mounting base and the camera lens. The three black bars are the actual bolts mounting the mirror above the camera and the reflections of these bolts.

The images produced by the camera system contain the full 360 degrees

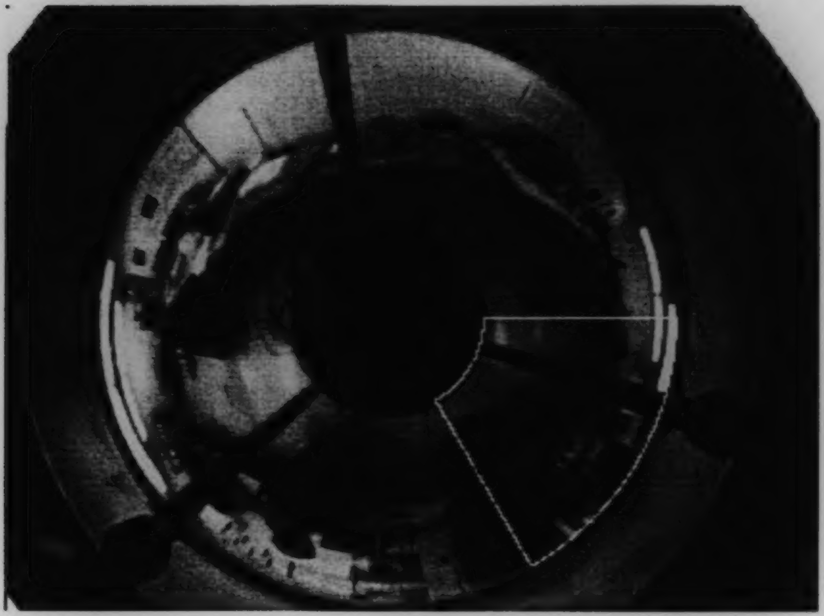


Figure 3.7: Full image frame from panoramic imaging system. Image resolution is 640x480 pixels. The white lines outline the section of image that will be warped into Figure 3.8. The outer rings in the image are part of the mounting system for the mirror. The next inner ring is the actual panoramic information, while the inner-most rings are reflections of the mounting base and the camera lens. The three black bars are the actual bolts mounting the mirror above the camera and the reflections of these bolts.

Figure 3.8: Warped image fragment from Figure 3.7. Physical image resolution is 320x240 pixels, warped from wedge fragment, outlined in white in Figure 3.7, with average resolution 240x180 pixels.

of panoramic information, but are more complicated and difficult to use than images produced by a standard, limited field-of-view camera. It is necessary to warp these panoramic images into a more intuitive form. The warping process takes an image provided by the camera system and, based on the point-of-view, field-of-view and aspect ratio, creates flat, limited field-of-view images that could have been taken from a panning camera system. Figure 3.8 shows an example of a section of the panoramic image that has been warped into a flat image. This image fragment is from the lower right hand corner of the panoramic image in Figure 3.7, outlined with the white lines. The black column that appears in the warped image fragment is from the mounting bolts used to hold the mirror above the camera.

The low resolution of the resulting images can be easily observed in Figure 3.8. The original wedge is 60 degrees, one sixth of the panoramic image, and has varying resolution at different distances from the centre. This gives a resulting unwarped image with different resolutions along horizontal lines. For example, at the top of the resulting image, a scan line consists of approximately 240 pixels from the outermost radius in the original image. For comparison, the bottom of the unwarped image only contains about 80 pixels from the panoramic image with the remaining 160 are interpolated pixels. The wedge in the original image has an average resolution of 240 × 180 pixels for the 60 degree wedge with the width of the wedge increasing with the distance from the centre of the panoramic image.

3.4 Summary

This chapter has presented the telerobot platform that has been used in the rest of this dissertation. It has discussed the hardware used for the robot as well as its construction. A brief discussion of the programming environment provided by the microcontroller has also been presented. Using the hardware and the programming environment, the robot is able to move forward and backward as well as turn in either direction using two different pivot points.

This chapter has also discussed the panoramic video system that can be installed on the telerobot when panoramic images are desired. These panoramic images can be unwarped to provide a simulated limited field-of-view image for an operator with reduced resolution. Chapter 4 will discuss the model of the panoramic video system and a more detailed examination of the unwarping process. It will also discuss an extension to the panoramic system to provide stereo images.

Chapter 4

Model for Panoramic Video

In the previous chapter, the telerobot hardware and video systems were discussed. The mathematically more interesting video system is the panoramic system. The images from this system are not immediately usable by a human operator since they contain a 360 degree view of the environment local to the telerobot. In order to present these images to the operator, they need to be processed into a more intuitive form.

This chapter will discuss the mathematical model of the panoramic system in Section 4.1. This model will enable the discussion, in Section 4.2, of the warping process used to process the images. Finally, Section 4.3 will discuss an extension to the panoramic video system that allows stereo images to be generated and depth information calculated.

4.1 Mathematical Formulation for Panoramic Video System

The panoramic video system consists of a hemispherical mirror mounted a known distance above an NTSC camera. To enable the mathematical discussion, define the coordinate system so that the focal point of the camera is at the origin (in 3–space) and the viewing axis pointing along the y-axis (see Figure 4.1). The image plane is located a distance f below the focal point (it can be considered to be a sub-plane of the plane $y = -f$). The mirror will be mounted such that the lowest point on the mirror is at distance d from the focal point. For uniformity of the resulting image, the mirror must be designed

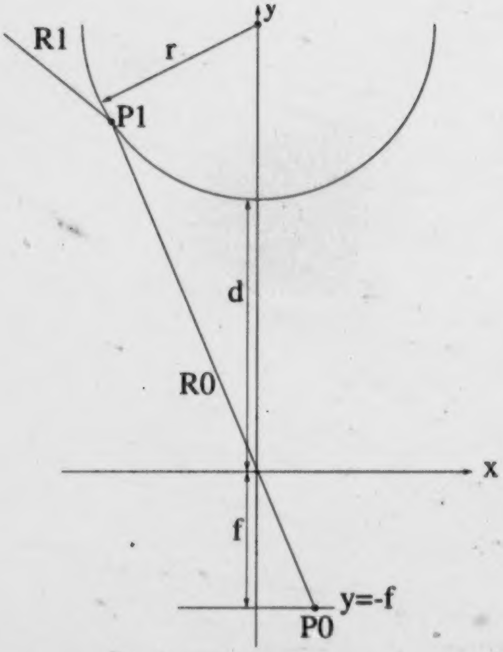


Figure 4.1: Single lobed mirror configuration in panoramic video system. The image plane is at $y = -f$, the focal point is at the origin $(0, 0)$, and the mirror, of radius r, is located at distance d from the origin. A point $P_0 = (x_0, -f)$ is produced from a point in space along ray R_1 using the ray R_0 $y = \frac{-fx}{x_0}$, intersecting the mirror at $P_1 = (x_1, y_1)$.

so that it is symmetrical about the y-axis. This allows us to work in only two dimensions (instead of the normal three).

Initially, consider a single lobed mirror which is defined as a part of the sphere:

$$x^2 + y^2 + z^2 = r^2 \tag{4.1}$$

Under the initial conditions (described above and shown in Figure 4.1), we only need to consider a circle in two dimensions. Also, since the surface is mirrored, and a light ray cannot pass through the surface, we only need to consider the lower half of the surface. This gives us the equation of a curve that represents our mirrored surface in the coordinate system defined above:

$$x^2 + (y - (d + r))^2 = r^2 \tag{4.2}$$

Now that the model has been constructed, we can determine the correspondence between a point on the image plane and locations or objects around the mirror. Initially, we can generate a ray into the surrounding space that corresponds to a given point on the image plane.

First, we determine the ray, R_0, that corresponds to a point on the image plane. Given a point $P_0 = (x_0, -f)$ on the image plane, P generates a ray R_0 passing through the origin to intersect the surface of the mirror. The equation of ray R_0 is:

$$y = \frac{-f}{x_0} x \tag{4.3}$$

This will intersect the surface of the mirror at some point $P_1 = (x_1, \frac{-f}{x_0} x_1)$ which satisfies:

$$x_1^2 + (\frac{-f}{x_0} x_1 - (d + r))^2 = r^2 \tag{4.4}$$

giving the solutions:

$$x_1 = -\frac{x_0(fd + fr - \sqrt{f^2r^2 - x_0^2d^2 - 2dx_0^2r})}{x_0^2 + f^2} \tag{4.5}$$

$$x_1 = -\frac{x_0(fd + fr + \sqrt{f^2r^2 - x_0^2d^2 - 2dx_0^2r})}{x_0^2 + f^2} \tag{4.6}$$

The solution 4.5 can be discarded since it represents the upper half of the circle defining the mirror (R_0 must pass through the bottom half of the mirror first, which it cannot do since it is a fully reflective surface).

For there to be a solution to equation 4.6, the following equation must be satisfied:

$$|x_0| \leq \frac{fr}{\sqrt{d^2 + 2dr}} \tag{4.7}$$

The case of equality in Equation 4.7 is the degenerate case of R_0 being tangential to the mirror at point P_1.

Ray R_0 in equation 4.3 will reflect about the normal to the curve defined in Equation 4.2, the mirror, at point P_1 and generate another ray, R_1. The first object that intersects with this reflected ray R_1 will appear on the image plane at the point P_0.

This reflected ray R_1 has the equation:

$$y - y_1 = \frac{2x_1^2(d+r) - y_1(y_1 - (d+r))^2 - x_1^2 y_1}{x_1(d+r)^2 - x_1 y_1^2 - x_1^3}(x - x_1) \tag{4.8}$$

We can also determine where a point in space will be projected on the image plane. A point in space $P_0 = (x_o, y_0)$ projects on to the image plane at a point $P_i = (x_i, f)$. The ray from P_0 intersects the mirror at the point $P_1 = (x_1, y_1)$. A ray from the point P_1 to P_i passes through the focal point at the origin O. From these 3 points: P_0, P_1, and O, we have two rays that intersect at P_1. At this point, the two rays must have an equal angle from the tangent to the mirror for this to be a solution for the point P_i.

For the spherical mirror, the point P_1 must satisfy:

$$y_1 = (d+r) - \sqrt{r^2 - x_1^2} \tag{4.9}$$

If we shift the coordinate system so the origin is at P_1, the tangent to the circle which represents the mirror is:

$$y\prime = \frac{x_1}{(d+r) - y_1} x\prime \tag{4.10}$$

The incoming ray (from P_0 to P_1) has the equation:

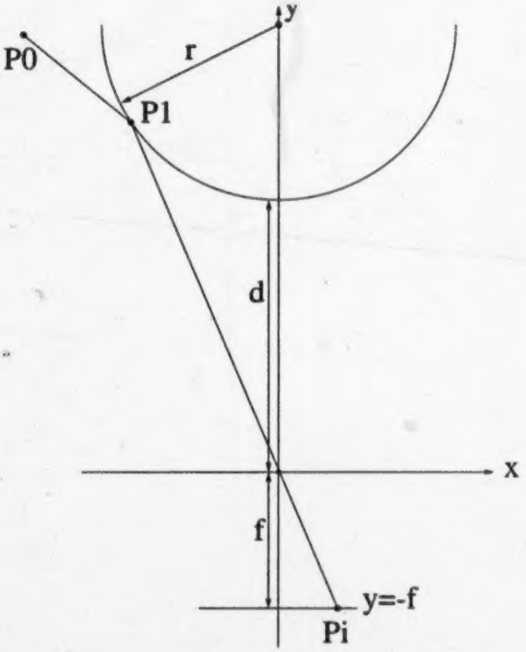


Figure 4.2: Single lobe configuration for calculating projection of a point in space $P_0 = (x_0, y_0)$ to the image plane producing point $P_i = (x_i, -f)$ intersecting mirror at point $P_1 = (x_1, y_1)$.

$$y' = \frac{y_1 - y_0}{x_1 - x_0} x' \tag{4.11}$$

The outgoing ray (from P_1 to O) has the equation:

$$y' = \frac{y_1}{x_1} x' \tag{4.12}$$

In order for P_1 to be a valid mirror point, the angle between the tangent and the incoming ray, θ_0, and the angle between the outgoing ray and the tangent, θ_1, must be equal.

$$\cos\theta_0 = \frac{(x_0 - x_1)((d+r) - y_1) + x_1(y_0 - y_1)}{\sqrt{((d+r) - y_1)^2 + x_1^2}\sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2}} \tag{4.13}$$

$$\cos\theta_1 = \frac{x_1((d+r) - y_1 + y_1^2)}{\sqrt{x_1^2 + y_1^2}\sqrt{((d+r) - y_1^2) + x_1^2}} \tag{4.14}$$

For a valid mirror point, $P_1 = (x_1, y_1)$ must satisfy:

$$\frac{(x_0 - x_1)((d+r) - y_1) + x_1(y_0 - y_1)}{\sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2}} = \frac{x_1((d+r) - y_1 + y_1^2)}{\sqrt{x_1^2 + y_1^2}} \tag{4.15}$$

4.2 Reconstruction from Panoramic Images

The ideal reconstruction technique would take the point in the image and determine a point in space that corresponds to a specific distance which will be constant for all points. Using this technique, we can map the image to a three dimensional surface and project back onto a flat viewing plane. Unfortunately, this is a complex calculation. For a single point on the surface, the sum of the distances along ray R_0 from the origin to P_1, using equation 4.6, and along ray R_1, using equation 4.8, must be a constant value. This generates a curved surface parameterized by the location on the image plane which then needs to be projected onto a flat viewing plane.

The ideal reconstruction process can be quite complex using the equations described previously. These can be solved once and a look-up table employed during the warping process. However, the solution can also be simplified by making some assumptions and simplifications. The first simplification is to map the image plane onto a cylinder (a vertical line, $x = x_c$, in the two dimensional case described in the previous chapter). This provides a simple surface that can be mapped onto a viewing plane easily. However, the vertical distance between points corresponding to consecutive image plane points is not constant. If we assume that it is, the mapping process is simplified still further and the resulting image is still viewable by a human operator.

The warping process used is a simple transformation to keep the processing time small. The warping translates lines of a constant angle (radiating from the centre of the panorama) in the original image to vertical lines in the target image; and lines of constant radius into horizontal lines. Assuming that the center of the circles representing the panoramic information is at $x = 0, y = 0$, the inside radius is r, and the outside radius of the panoramic circle is R, the pixel intensity for a given pixel at (u, v) in the target image is the same as the intensity of the pixel closest to (x, y) in the panoramic image where:

$$x = (r + \frac{v * (R - r)}{L}) * \cos(\theta + \frac{u * \phi}{W})$$
$$y = (r + \frac{v * (R - r)}{L}) * \sin(\theta + \frac{u * \phi}{W})$$

where W and L are the width and height of the target image, θ is the angle corresponding to the right edge of the image fragment, and ϕ is the angle encompassing the whole image fragment from right edge to left edge.

Since the warping equations are non-linear, the scan lines in the final warped image vary in quality. The bottom scan lines are derived from smaller circles in the original image, with fewer pixels of data than the upper scan lines, which are generated from larger circles. This results in images that have varying quality from top to bottom. The top scan lines are crisper than the bottom lines, as we can see in Figure 3.8.

Viewing the full 360 degree panoramic image, either in its original unwarped form or completely expanded, can make operating a telerobot very difficult – especially with the quality provided by our camera system of 640 × 480 pixels. In order to make the display meaningful for the operator, and less confusing, we need to display only a portion of the panoramic image. This image

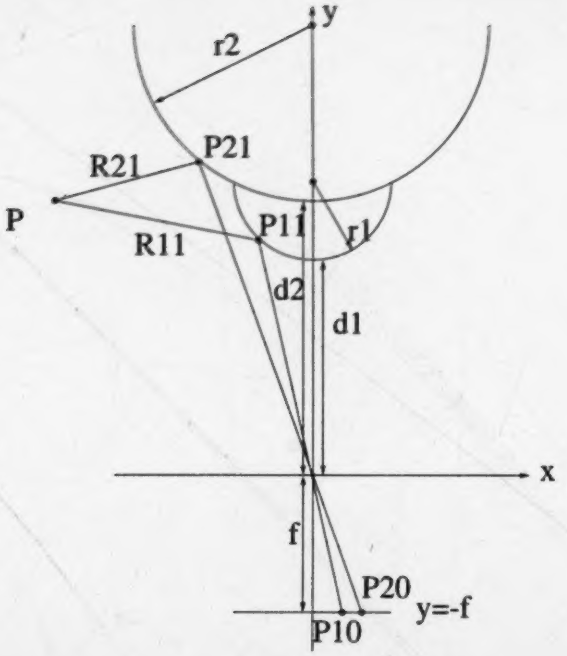


Figure 4.3: Double lobed mirror configuration for panoramic stereo. The image plane is at $y = -f$, the focal point is at the origin $(0,0)$, the smaller lobe, radius r_1, located distance d_1 from the origin, and the larger lobe, radius r_2, located at distance d_2 from the origin. The points $P_{10} = (x_{10}, -f)$ and $P_{20} = (x_{20}, -f)$ each produce rays through the origin which intersect the mirror lobes at $P_{11} = (x_{11}, y_{11})$ and $P_{21} = (x_{21}, y_{21})$ respectively and meet at a point in space $P = (x, y)$.

fragment is determined by moving a viewing window around the panoramic image and only displaying the fragment that can be seen through this window. Each fragment is warped from a wedge which, for a 60 degree field of view, averages 240x180 pixels. The operator can move the window around the panoramic image using the mouse. By predicting the mouse motion, using a predictive Kalman filter, we can transmit not only the portion of the image currently being viewed, but also the portion that the operator is predicted to be viewing in the near future. This will be discussed further in Chapter 6.

4.3 Panoramic stereo

The single lobed mirror used in the panoramic system described above can be replaced with a double lobed mirror (originally presented in [71]) to provide a means of determining depth information. This information could be used in either a more accurate reconstruction of images from panoramic video, determining motion of objects, or mapping the environment for the telerobot in semi-autonomous tasks. The panoramic video system includes a second mirror that is double lobed. This subsection provides the theoretical formulation and some simple experiments that were conducted in this area. Unfortunately, the speed and accuracy of the approach show promise, but are not of sufficient quality at this time to use in the teleoperation experiments.

4.3.1 Theoretical Formulation for Panoramic Stereo

For this theoretical formulation, the same assumptions as in a single lobe mirror case are used. For a double lobed mirror (see Figure 4.3, with lobes parameterized by r_1, d_1 and r_2, d_2, a point in space can be completely determined from two points in the image plane $(x_{10}, -f)$ and $(x_{20}, -f)$. The two points determine two mirror intersection points (one for each lobe) (x_{11}, y_{11}) and (x_{21}, y_{21}). The intersection of the two reflected rays, R_{11} and R_{21}, provides the point in space that corresponds to the two points in the image plane. The distance from the x coordinate of the intersection point provides the distance to camera axis for the point. We can also determine the Euclidean distance of the point to the focal point in the standard way.

The point in space $P = x, y$ satisfies the two equations:

$$y - y_{11} = \frac{2x_{11}^2(d_1 + r_1) - y_{11}(y_{11} - (d_1 + r_1))^2 - x_{11}^2 y_{11}}{x_{11}(d_1 + r_1)^2 - x_{11}y_{11}^2 - x_{11}^3}(x - x_{11}) \tag{4.16}$$

$$y - y_{21} = \frac{2x_{21}^2(d_2 + r_2) - y_{21}(y_{21} - (d_2 + r_2))^2 - x_{21}^2 y_{21}}{x_{21}(d_2 + r_2)^2 - x_{21}y_{21}^2 - x_{21}^3}(x - x_{21}) \tag{4.17}$$

with the added constraints:

$$y_{11} = \frac{-f}{x_{10}} x_{11} \tag{4.18}$$

$$x_{11} = -\frac{x_{10}(fd_1 + fr_1 \pm \sqrt{f^2r_1^2 - x_{10}^2d_1^2 - 2d_1x_{10}^2r_1^2}))}{x_{10}^2 + f^2} \tag{4.19}$$

$$y_{21} = \frac{-f}{x_{20}}x_{21} \tag{4.20}$$

$$x_{21} = -\frac{x_{20}(fd_2 + fr_2 \pm \sqrt{f^2r_2^2 - x_{20}^2d_2^2 - 2d_2x_{20}^2r_2^2}))}{x_{20}^2 + f^2} \tag{4.21}$$

4.3.2 Preliminary Results for Depth from Panoramic Stereo

In order to conduct experiments with panoramic stereo, the size of the mirror lobes must be determined. This is either known from manufacturing process or can be physically measured. This gives the radius of the two lobes and an offset of the centres of the smaller lobe from the larger one. This means that we only need to determine one distance (either the height of the bottom of the smaller lobe or the height of the centre of the larger lobe) rather than both distances.

To determine the height of the lobes and the focal length f, the distance from the origin to the image plane, we use the principle of similar triangles. We need to take two pictures of an object with known length, l, positioned at a known height, h, from the camera. This object projects into a measurable image. The object placed at height h may not be exactly that distance from the origin. This means we need a small offset, o, which is added to any height measurement from a known plane (in our case, the plate which the camera is mounted to) to get the actual height values. Using the two calibration pictures and ratios from similar triangles, we can get a value for f, 618 pixels, and the offset, $o = 0.0914$ cm. Note that these two values use different units, pixels and cm. This is not a problem since all values below the origin are calculated in pixels which is proportional to cm, which is in measurements for all values above the origin, and the constant of proportionality does not affect the equations (it cancels from the equations and only the ratio is important).

We have measured the height of the centre of the large mirror lobe from the camera mounting plate. Using this value and the offset determined above, we can get the geometry of the camera and mirror system: $r_1 = 1.94$ cm, $r_2 = 5.78$ cm, $d_1 = 8.75$ cm, $d_2 = 10.3$ cm, and $f = 618$ pixels.

Several experiments were conducted using different images. The best results were obtained using images of lights. These produced the best results because they provided the easiest estimates of matching points in the two images. Using bright lights allows easier matches.

To be able to conduct the experiments the actual position of the object is measured from the centre of camera mounting plate (which is offset by the value of o determined above). Once we have actual positions, we can try to determine the value using the position equations.

The first step in the calculation process is to identify an initial set of estimates for the matching points. Using these initial matches, we calculate the position of the object and refine our initial match. This refinement process is necessary because of the poor resolution of the images (especially from the smaller mirror) and because we are trying to show that we can get reasonable position results, not that the matching process works well (we are matching manually, not automatically). Since the equations are based on radial symmetry, once the initial matches are made, we are only working with the radial distance from the centre of the image.

Experiments were conducted using two different light positions. The results are presented in Tables 4.1 and 4.2 with the horizontal radial distance (actual and estimated), the vertical height from the plane of the origin (actual and estimated), and the Euclidean distance from the origin (actual and estimated).

Tables 4.1 and 4.2 show that the horizontal distance can be estimated accurately, within one percent. Unfortunately, the estimate of the vertical distance is considerably worse. The estimated distance has an error of approximately 14 to 17 percent. This is most likely due to error in the matching process.

4.4 Summary

This chapter has presented the mathematical model of the panoramic video system. This has shown that a reflecting ray can be calculated for every pixel

distance	actual	estimate
horizontal	108 cm	106.976 cm
vertical	54 cm	46.584 cm
Euclidean	120 cm	116.679 cm

Table 4.1: For Light 1, radial distance in image: $x_{10} = 27$ pixels, $x_{20} = 168$ pixels

distance	actual	estimate
horizontal	215 cm	215.878 cm
vertical	43.5 cm	51.383 cm
Euclidean	219 cm	221.909 cm

Table 4.2: For Light 2, radial distance in image: $x_{10} = 93$ pixels, $x_{20} = 200.5$ pixels (estimated).

in an image and the pixel will show the first object encountered along this ray. This process can also be reversed determining which pixel corresponds to a specific point in space around the mirror. A similar model for a stereo panoramic system can be used to reconstruct depth information about a scene.

The model of the panoramic video system can be used to reconstruct the scene and project it onto a window showing only a small segment of the image. Using simplifications, the speed of reconstruction process can be increased so that it can be accomplished at a rate enabling real-time viewing of a scene.

The following chapter will present the operator interface and supporting software. The interface can present the operator sections of the panoramic video stream or standard NTSC video images. The interface is divided into two program suites: the server suite which runs on the base station, and the client suite supplying the interface to the operator.

Chapter 5

Teleoperation Interface Software

The teleoperation software has undergone several incarnations over the course of the research program, but can be separated into two basic programs. The first interface program runs on a single SGI Indy. This program is set up to display image frames as they are registered by the video server and send commands to the robot using the numeric keypad. The program only displays images from one of the cameras because the Indy can only accept one NTSC video sequence. This program is used to demonstrate a simple telepresence interface where the base station is used as the operator terminal.

The second interface is actually two suites of programs (see Figure 5.1). It is designed using a client/server model. The server suite is run on the base station which is connected to the telerobot. This suite accepts requests from the client suite and returns information or changes the telerobot motion. The client suite is the operator interface. This suite displays the images as they are received from the server and also takes operator input to change the telerobot's motion or change parameters for the image processing modules.

This chapter will describe in more detail the second suite of programs. First, Section 5.1 will describe the server suite of programs. The client suite will then be discussed in Section 5.2.

5.1 Server Suite

The server suite of programs consists of a video server, a robot server and a communications server. These server subprograms communicate amongst each

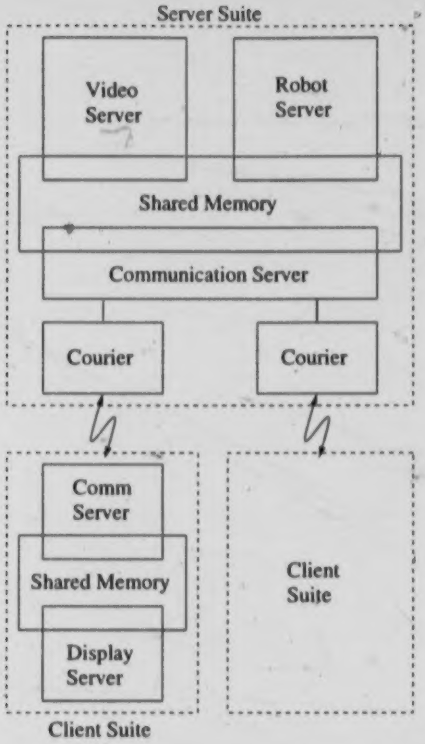


Figure 5.1: Software block diagram

other using shared memory. The video and robot servers can be voluntarily turned off when the server suite is started. Communication between client and server was originally designed to allow multiple clients to connect to the server, but this capability was removed to keep the server processing speed as high as possible.

The server suite has been used on several different platforms. Early experiments used an SGI Indy as the base station. This platform provided reasonable performance using the vl library functions and built in video hardware. The suite was also used on an SUN Sparc Ultra-1 without the video server at the official opening ceremonies for the Research Institute for Multimedia Systems. The server suite has also been modified for use under the RedHat Linux kernel version 2.2.14. This final base station is the platform used for the experiments using feature images and hybrid video as well as the quantitative measures of performance.

5.1.1 Shared Memory

Communication between processes in the server suite is accomplished using shared memory. The shared memory contains the most recent video frames captured as well as information regarding the robot motion and which frame is being processed by other subsystems.

The video frames are placed into one of several buffers depending on which is free (not being used by the communication server). There are two counters to indicate which frame is being read, if any, and which frame can be read next. In order to eliminate the problem of updating video frames while it is being used, which leads to a view from sections of different image frames, the shared memory can contain a number of previous image frames. Currently, the shared memory is set to store two frames at a time. This can be increased if stale data is being viewed. Two frames provide a reasonable video stream for viewing.

The shared memory also contains information about the robot motion. When a request to change the motion is received from the communication server, it places the new motion commands in the shared memory segment and the robot server can then receive this information and notify the robot of the requested change in motion.

5.1.2 Robot Server

The robot server is a simple process that communicates with the serial port of the base station in order to pass information to the microcontroller on the telerobot. This server module reads the motion command from the shared memory and, if it is different from the last command sent to the robot, informs the microcontroller of the motion change. This module can be expanded to include more sophisticated commands than are currently supported.

5.1.3 Video Server

Video capture is dependent on the hardware and operating system used on the base station. An SGI base station uses a different video library than a Linux base station. Since either platform can be used as a base station, this module of the server suite has two forms. Both forms follow a similar structure, but with some differences.

The main structural difference in the video servers is how the images are retrieved. Using the vl library on the SGI, once the video stream is initialized, an event loop is entered. This video event loop is defined by the library and my video server provides a procedure to call when a video event is detected. This event loop can also handle gl events when the video server and interface are components of the same program, as in the very first interface written. When the event loop detects a video event indicating a new frame has been captured, it processes the image frame. The processing of a new frame is the same on either base station.

The video server on a Linux base station does not use an event based loop. For this version of the server, a polling loop is used. The initialization of this video stream includes defining two frames to effect double buffering allowing faster processing by requesting one video frame while processing the second. Once the initialization is complete, the server enters an infinite loop. In this loop, it requests the next frame be captured by the video capture card and processes the previously requested frame.

The processing of each frame is the same for either version of the video server. It is also very simple as the video server simply copies the recently captured video frame into the shared memory segment. The frame is copied into the next frame buffer in the shared memory segment that is not being read or processed by the communication server. Each image is stored as raw data similar in form to a PGM image, an 8-bit per pixel greyscale image.

The video stream is captured independently of the rest of the server suite. This was done to eliminate possible bottlenecks in requesting new images and having to capture them each time a request is made by a client. The capture process will continually repeat until the video server is terminated by the controlling process of the server suite.

5.1.4 Communication Server

The original server design allowed multiple clients to connect to the server suite at any given time. This would allow more than one operator to share control of the telerobot. This capability was removed to eliminate the problems inherent with sharing control among more than one operator and to improve performance. The result was to allow only one client to be connected at a time, but any number of clients could connect in any order. The communication server controls which client is controlling the telerobot. It is also the controlling process for the server suite. When the communication server terminates, so do both the video server and the robot server if either is executing.

When a client connects to the communication server, using a TCP socket, the server stops listening for new communication requests and responds only to this new client, as a courier process. Originally, this would spawn a courier process which would respond only to this client while the communication server continued to listen for new requests and spawn a new courier process for each client.

The courier process is the heart of the server suite. It is this process which gets requests from the client, processes them, and returns the results. This process performs any image processing necessary, for example edge detection and compression or clipping, to reduce the transmitted image sizes, and transmits robot motion changes to the robot server via shared memory. Other requests include changes to the image processing parameters and terminating the current client session by halting the robot.

The major request that the couriers must deal with is the request for image data. Figure 5.2 shows the processing involved with an image request. Based on the type of request (standard image, edge image, panoramic image, etc.), the courier may process the most recent frame stored in the shared memory segment. These image processing routines are designed so that they may be changed as new procedures are implemented. It is also designed to perform

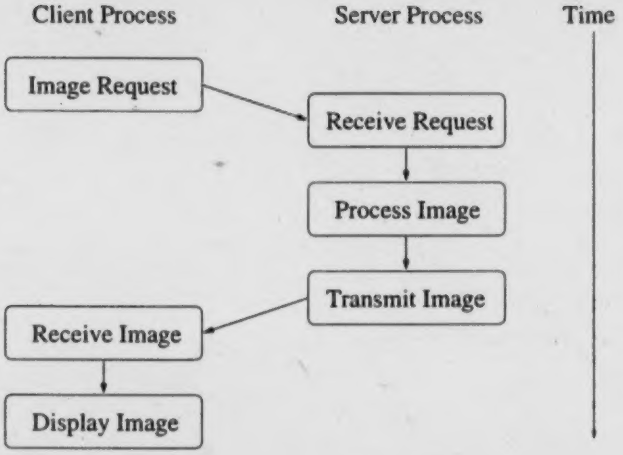


Figure 5.2: Image request communication process. The client sends an image request to the server. The server processes and returns the image, which the client then displays.

the image processing on either limited field-of-view images or panoramic images. The possible image processing techniques are discussed in more detail in Chapter 7.

5.2 Client Suite

The client suite only contains two programs. The first is the display process which controls the X display. It takes images from shared memory and puts them into the window structure for the X server. It also puts robot control requests into shared memory for the communications process to manipulate. The other process in the client suite is the communications process. This process takes requests from shared memory and sends them to the server communications process.

The client suite has two interface types. The first interface uses the X11 library functions and therefore is usable on any platform which uses X Windows. Unfortunately, this interface does not provide a sufficiently fast display rate as we can see from the discussion of the display client in Section 5.2.3. The second interface is written using SGI's GL library. This provides a much better display rate than the X interface. Since this is the preferred display method and due to the limited number of SGI's available in the lab, the final configuration of the interface programs is to have the server suite running on a Linux workstation while the client interface is running on the Indy.

5.2.1 Shared Memory

As in the server suite, communication among the different processes in the client suite is accomplished using shared memory. Unlike the server suite, the shared memory in the client can be quite large. Not only does it contain several video frames and robot motion commands, but it also includes different types of video frames (full video and feature images), parameters for image processing (edge detection threshold, segmentation thresholds, etc.) and parameters for determining subregions (viewed and predicted for limited field-of-view and panoramic).

Different operating systems place different limits on the maximum size of the shared memory segments. This can limit the usefulness of the client suite. For example, under Solaris, the shared memory segment cannot contain more than a single image frame and a single feature image. This means that we encounter discontinuities when viewing the resulting video sequence. These discontinuities appear when the image frame is modified by the communication process while the display is processing the frame. This generates images that are composed of more than one actual video frame.

Just as in the server suite shared memory, the client suite attempts to maintain previously viewed frames in shared memory. The same reasons apply to the client as they did for the server suite. In addition to the standard video frames, the client also maintains a set of feature image frames, called overlays, in the shared memory segment. Because the client suite can request and display both standard video frames and the overlay frames (see the discussion of Hybrid Video in Chapter 8) this shared memory segment must contain a history of both types of video frames.

The remaining information stored in the client shared memory segment includes: image processing parameters, subimage regions, robot motion commands, motion estimation parameters, and frame rate information. This information is used by both the communication and display processes and is placed in shared memory so either process can access it at the same time. Only one process will change any given portion of this shared memory, but both require access.

5.2.2 Communication Client

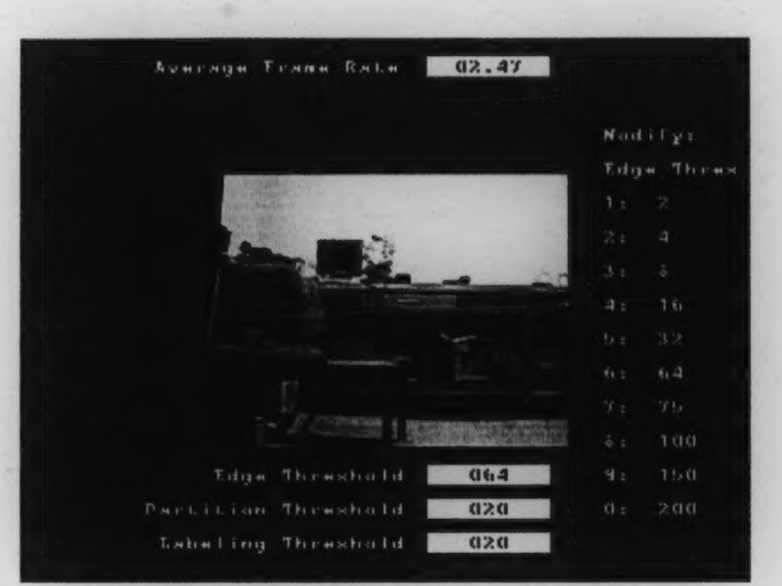
The communication client process performs all communication with the server suite. It uses standard TCP socket routines and follows a request/receive protocol. The communication process requests image frames whenever it can to provide the operator with the most up to date images possible. If there are other requests that are needed, motion changes or image processing parameter changes, it interleaves these requests with the frame requests.

When an image frame is requested, the communication process includes the subregion being requested and the image type (limited field-of-view, panoramic, edge, etc.) in the request. The server processes the request and transmits the requested image. The exact subregion could be modified if the server cannot transmit exactly what is being requested. Once the image has arrived at the client, it is processed as needed (unpacked for edge images, unwarped for panoramic) and placed into the shared memory segment for the display process.

5.2.3 Display Client

The display client process is the actual interface with the operator. This processing is separated from the server communication to eliminate the possibility of a bottleneck in either process from interfering with the other process. This allows the communication with the server to be independent from the communication with the operator.

The two versions of the client, X11 vs. GL, are very similar in processing. The only difference is the library functions used and the display performance. There is a difference in how the interfaces appear due to the different libraries, but the operation is the same.


Average Frame Rate 02.47

Modify:
Edge Thres
1: 2
2: 4
3: 8
4: 16
5: 32
6: 64
7: 75
8: 100
9: 150
0: 200

Edge Threshold 064
Partition Threshold 020
Labeling Threshold 020

Figure 5.3: X11 based operator interface.

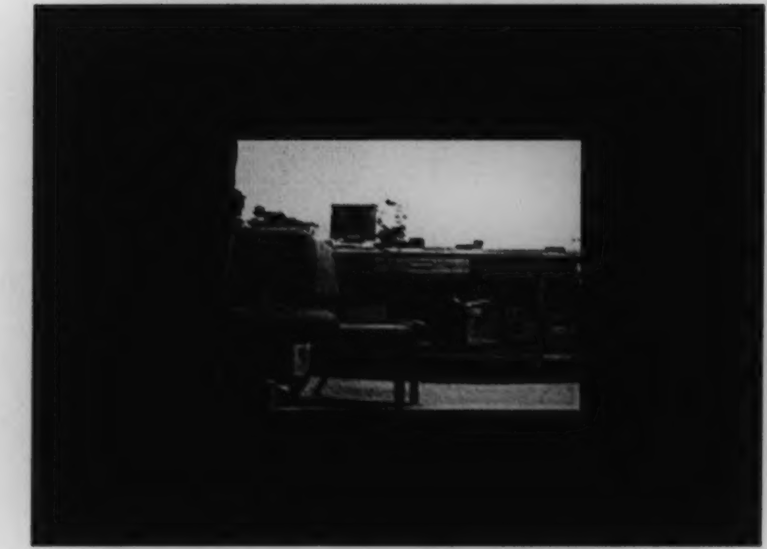


Figure 5.4: GL based operator interface.

There is another obvious difference in the contents of the two displays, shown in Figures 5.3 and 5.4. The X11 display provides some text information relating to the frame rate, values of current image processing parameters, and possible values for one of the processing parameters. This information was removed from the GL display since, while helpful, it is unnecessary for performing the experiments. This textual information does not take significant amounts of processing time to display and is not considered in the performance comparison of the interfaces.

A comparison of the two interfaces can be done using the *display rates* of each. The display rate is calculated as the number of times per second the image subwindow, the central portion of the interface, is updated. The X11 based interface only updates the image subwindow approximately 8.09 times per second due to the structure of X requests. The client must communicate with the X server and request to place an image into the window. This communication is additional overhead not present in the GL version of the interface.

In contrast, the GL version updates the image 28.25 times per second. This library has a more direct connection to the graphics hardware installed in the Indy. The GL library places the image information directly into the display buffer without additional communication to a server process. This shows that the GL version can provide a more realistic interface, especially when the video stream is transmitted at 10 frames per second or higher.

The interface allows the operator to control the robot motion using the numeric keypad. The operator can move the robot forward or backward, it can turn sharply or slowly to the left or right, or the operator can stop the robot. The motion is controlled using the keys as shown in Figure 5.5. These possible motions correspond to the motions programmed into the microcontroller as described in Section 3.2.

The operator can also modify the image processing parameters. There are several preset values for the edge threshold used in the Sobel edge detection algorithm. The operator can choose between these values using the number keys. There are also values for the image segmentation algorithm which was used in

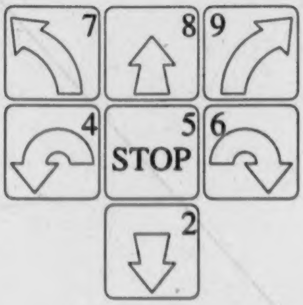


Figure 5.5: Numeric keypad and robot motion. The eight and two keys move forward and backward. Four and six rotate left and right. Seven and nine turn gently left and right.

some of the preliminary work. These image processing procedures remain in the interface although they are not used in the hybrid video experiments.

The operator controls the viewing direction using the mouse. Using a click and drag approach, the operator can move the viewing window around and see different parts of the scene. When the panoramic video system is being used, the operator can view the full environment around the telerobot by dragging the video frame to the left or right and change the viewing angle. This aspect is discussed in more detail in Chapter 6.

5.3 Summary

This chapter has presented the interface programs used in this research. They can be separated into two groups: the server suite executing on the base station, and the client suite used in the operator interface. Each of these uses shared memory to communicate between components. Communication between suites is accomplished using TCP sockets.

Chapter 6 describes the experiments using predictive displays. The interface allows the operator to view only a small section of the images while the prediction process determines the motions that the operator will use to view other sections of the images. This chapter will discuss the models used in the prediction and present several experiments that demonstrate the effectiveness of the prediction.

Chapter 6

Prediction

The first step in reducing the amount of data transmitted is to reduce the size of the viewed image. In a panoramic image, we could show the full 360 degree image, but that image is difficult to interpret by a human operator. We are used to viewing limited field-of-view images. We can reduce the amount of data transmitted from the server by only transmitting a section of the full image. This will produce an image that a human operator can easily use.

Since the panoramic image contains more information than we are transmitting at a time, the operator may want to move the point of view so that he can view different sectors of the surrounding environments. When the operator makes a point of view change, there is a delay between when the operator makes the change (in the interface) and when the images will change. This update rate is affected by both the latency and the limits imposed on the available bandwidth. If the image being transmitted is large compared to the available bandwidth, the time taken to transmit the new image can cause a noticeable delay in the update of the display.

In order to overcome the delay in fetching new images from a different angle of view, we can transmit a wider section of the panoramic image. There are several methods that can be used to decide what sections to transmit: a constant amount beyond the viewing section on both sides of the image, a constant amount beyond the viewing section on the side of motion, or an amount based on previous motion. We use the third method to determine what sections to transmit.

Since the operator uses a mouse to control the viewing direction, we predict how the mouse is being moved. The prediction, using a Kalman filter, is used to determine where the operator is going to be viewing during the next small amount of time, the time to transmit two image frames. This allows us to reduce the apparent delay caused by changing viewing directions and also reduce the amount of data transmitted so only a small amount of unnecessary transmission is done.

This chapter presents research done using prediction to reduce the amount of data transmitted and also to reduce the apparent delay when the operator changes viewing direction in a panoramic telerobotic system. The first section, Section 6.1 will introduce Kalman filters and the models used in predicting mouse motion. This section will also show comparisons between the different mouse models and discuss the error covariance modeling used to improve the initial prediction results. Section 6.2 will describe the experiments conducted to show that using prediction improves the operator's experience and performance in the telerobotic tasks and discuss the results obtained.

6.1 Kalman Filters

Kalman filter algorithms have been used to solve problems in a wide range of areas. These areas include navigation in space, tracking objects using radar data, and robot localization using data from multiple sensors. Kalman filters can be used to derive an estimate of the state of a system based on a linear function of the previous states, which minimizes the error variance. Problems involving either continuous or discrete processes can be solved using Kalman filtering.

There are a number of different approaches to deriving the final equations used in a Kalman filter algorithm. Each derivation arrives at a slightly different set of equations used to estimate the state of the system; [12], [44], [61] and [62], all derive distinct sets of equations, but all will solve the same set of problems and generate very similar results. There are many different extensions that can be applied to the different algorithms, but the interesting extension for this work involves prediction of the system state instead of simple estimation of the current state. This extension of a standard Kalman filter algorithm is called a *predictive Kalman filter*.

One of the many uses of predictive Kalman filters is in predicting motion – for example, head motion in virtual reality [46], or trajectories for active tracking [14]. In this work, a Kalman filter tracks an operator's movement of a mouse, used to control the viewing window for a teleoperation display. This application requires a discrete filter approach since the sampling interval is non-zero. Application of the Kalman filter technique requires a mathematical model of the underlying dynamic system in terms of a set of state equations describing how the model changes over time. This model is assumed to be a first-order autoregressive process driven by zero-mean white noise. The prediction algorithm is trying to minimize the mean-square prediction error.

6.1.1 Predictive Kalman filter algorithm

The predictive Kalman filter algorithm used in this work is based on the approach described in [12] due to the relative simplicity of the equations. In this algorithm, the system being modeled can be described by a set of state equations of the form:

$$\mathbf{x}_{k+1} = \phi_k \mathbf{x}_k + \mathbf{w}_k \tag{6.1}$$

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k \tag{6.2}$$

where $\mathbf{x}_k$ represents the state at time interval k (a first-order autoregressive process), ϕ_k is the state matrix describing how the state changes from one time interval to the next, and $\mathbf{w}_k$ is random process noise with zero mean and covariance matrix $\mathbf{Q}_k$. The first equation gives us a value for $\mathbf{x}_{k+1}$ – the actual value of the model at time interval $k+1$ based on the previous state. $\mathbf{z}_k$ is the measured state in time interval k, $\mathbf{H}$ is the observation matrix which relates $\mathbf{x}_k$ and $\mathbf{z}_k$, and $\mathbf{v}_k$ is random measurement noise with zero mean and covariance matrix $\mathbf{R}_k$.

Once the state of the system can be described, the Kalman filter can be applied to predict the value of the system during the next time interval. The filter process solves the following set of equations:

$$\mathbf{K}_k = \mathbf{P}_k\mathbf{H}^T(\mathbf{H}\mathbf{P}_k\mathbf{H}^T + \mathbf{R}_k)^{-1} \tag{6.3}$$

$$\mathbf{P}_{k+1} = \phi_k(\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{P}_k\phi_k^T + \mathbf{Q}_k \tag{6.4}$$

$$\hat{\mathbf{x}}_{k+1} = \phi_k\hat{\mathbf{x}}_k + \phi_k\mathbf{K}_k(\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_k) \tag{6.5}$$

where $\mathbf{K}_k$ is the filter gain, $\mathbf{P}_k$ incorporates filter gain and the predicted covariance matrix, and $\mathbf{I}$ is the identity matrix. The predicted value of the state is $\hat{\mathbf{x}}_{k+1}$.

This set of equations allows one step prediction. That is, the algorithm can predict a single time step in the future. In order to predict several time steps, a modification of the algorithm is needed. In this application, the state matrix ϕ_k has elements that are based on the sampling time interval δt (see the next section for details). When predicting more than one time interval, a second value for ϕ_k is used, called ϕ_τ. The value of ϕ_τ is based on the larger time interval, τ. This gives the predicted state, $\hat{\mathbf{x}}$, which is returned by the predictive Kalman filter:

$$\hat{\mathbf{x}} = \phi_\tau\hat{\mathbf{x}}_k + \phi_\tau\mathbf{K}_k(\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_k) \tag{6.6}$$

The value $\hat{\mathbf{x}}$ is not retained however. This predicted value is only returned as a multiple time step prediction and is discarded. Since the prediction process is recursive and based on a single time step, the value $\hat{\mathbf{x}}_{k+1}$ is still calculated from equation 6.5 and it is this value that is used in the recursive algorithm.

6.1.2 Modeling Mouse Motion

The teleoperation interface uses mouse motion to allow the operator to control the viewing window. This allows the software to transmit only a portion of the full image while allowing the operator to choose where they are looking. To provide the operator with an appearance of continuity in the image sequence even when they are moving the viewing window, the system uses the predictive Kalman filter algorithm described above to predict the motion of the mouse.

The prediction allows the system to request image sections that contain more information than is currently being viewed to reduce the apparent delay while refreshing the image sequence without transmitting the full image.

The prediction process requires a model which can be defined as a set of state equations. The state vector must contain all of the variables to be predicted. In this system, the state vector contains the position, velocity, and acceleration of the mouse in both x and y directions. To simplify the discussion of the modeling process, a one-dimensional model of mouse motion along the x direction is used in this section. In the actual system both x and y dimensions are modeled in a similar manner. The two dimensions are assumed to be independent so motion in the x direction does not affect motion in the y direction. This simplifies the computations at the expense of the accuracy of the prediction.

The one-dimensional motion prediction state vector, in the x dimension, is defined as $\mathbf{x}_k = [x_k, \dot{x}_k, \ddot{x}_k]^T$, where x is the position of the mouse, $\dot{x}$ is the velocity, and $\ddot{x}$ is the acceleration.

An important part of the prediction process is to relate the actual state of the system, $\mathbf{x}_k$, with the measured state, $\mathbf{z}_k$. The measured state can be determined by the interface by determining the position of the mouse in the window. This position is requested from either the window manager or the graphics library, GL or X11. The velocity of the mouse can be calculated from the current and previous positions along with the time between samples. Both the position and velocity calculations are prone to error. The position error is based on the quantization of the pixels while the velocity error combines the quantization error with errors in time of measurement and calculation.

The last matrix to be calculated before discussing the physical model of the mouse is the observation matrix $\mathbf{H}$ relating the actual state with the measured state. The observation matrix is independent of the physical state. In this system, the observation matrix is simple as the measured state of the mouse and the actual state should only differ by the error process, $\mathbf{v}_k$. In the one-dimensional discussion of this section, the observation matrix is:

$$\mathbf{H}_k = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{6.7}$$

Since we wish to model mouse movements, we need to determine an appropriate physical model for the motion in terms of ϕ_k, $\mathbf{w}_k$, and $\mathbf{v}_k$. For ϕ_k, we experimented with three different forms of the state model. Each one is progressively less complex. The first model assumes that the mouse can be modeled as a point mass under the effects of a constant external force and damping, or viscous friction. For each direction of motion, we have the general equation of motion:

$$m\ddot{x} = F - b\dot{x} \tag{6.8}$$

where m is the mass of the mouse, F is the external force applied (assumed to be constant) and b is the coefficient of viscous friction. We call Equation 6.8 our *frictional model*.

Using this model with all the constants assumed to be non-zero, the initial differential equation that needs to be solved can be rewritten as $\ddot{x} = F_m - f_r\dot{x}$ where F_m is the constant force divided by the mass and f_r is the ratio of the frictional coefficient to the mass, which we call the *frictional ratio*. Solving this differential equation gives the following set of three equations for position, velocity and acceleration:

$$\begin{aligned} x(t) &= \frac{-ce^{-f_r t}}{f_r} + \frac{F_m}{f_r}t \\ \dot{x}(t) &= ce^{-f_r t} + \frac{F_m}{f_r} \\ \ddot{x}(t) &= -cf_r e^{-f_r t} \end{aligned}$$

Discretizing this set of equations, we derive the following set of equations as the state equations which are used in the Kalman filtering process:

$$\begin{aligned} \ddot{x}_{k+1} &= \ddot{x}_k e^{-f_r \Delta t} \\ \dot{x}_{k+1} &= \dot{x}_k + \ddot{x}_k \frac{(1 - e^{-f_r \Delta t})}{f_r} \end{aligned}$$

$$x_{k+1} = x_k + \dot{x}_k \Delta t + \ddot{x}_k \frac{(e^{-f_r \Delta t} + f_r \Delta t - 1)}{f_r^2}$$

This gives a state matrix for the *frictional model*:

$$\phi_k = \begin{bmatrix} 1 & \Delta t & \frac{e^{-f_r \Delta t} + f_r \Delta t - 1}{f_r^2} \\ 0 & 1 & \frac{1 - e^{-f_r \Delta t}}{f_r} \\ 0 & 0 & e^{-f_r \Delta t} \end{bmatrix} \tag{6.9}$$

If we assume that the frictional coefficient is zero but the external force is non–zero, then we have a *constant acceleration model.* The defining differential equation in this model is $\ddot{x} = a$ for some constant acceleration a. Such a model requires the same size of matrix as the friction model, but it is a simpler matrix, i.e., we have the state matrix:

$$\phi_k = \begin{bmatrix} 1 & \Delta t & \frac{1}{2}\Delta t^2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix} \tag{6.10}$$

The last assumption is that the external force is zero. This state is represented by the differential equations $\ddot{x} = 0$ and $\dot{x} = v$, where v is some constant value. We call this the *constant velocity model.* This is the simplest model that we looked at and reduces the size of the state matrix by 1 for each dimension modeled:

$$\phi_k = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \tag{6.11}$$

To characterize $\mathbf{w}_k$ and $\mathbf{v}_k$ in the environment, we assume that each type of noise is a random process with zero mean, and that the error covariance matrices $\mathbf{Q}_k$, for the process noise $\mathbf{w}_k$, and $\mathbf{R}_k$, for the measurement noise $\mathbf{v}_k$, are diagonal and of the form:

$$\mathbf{Q}_k = diag(q_k^2, q_k^2, q_k^2) \qquad \mathbf{R}_k = diag(r_k^2, r_k^2)$$

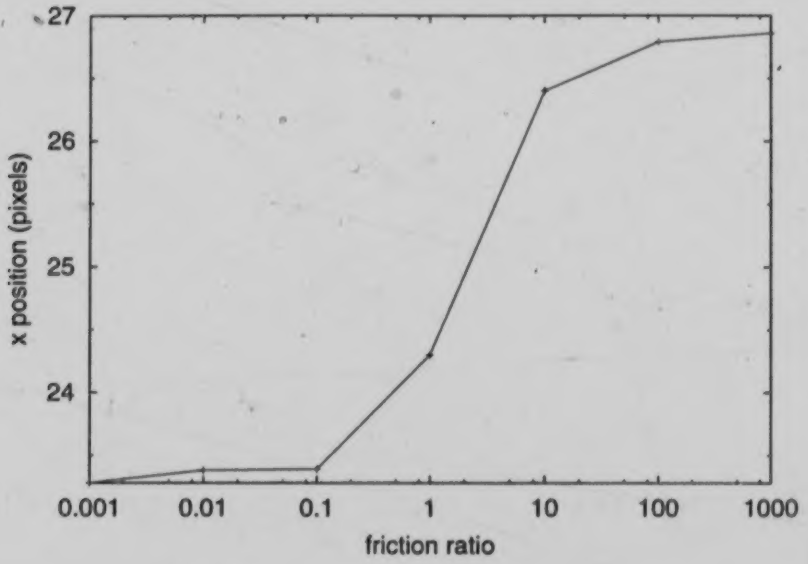
Identifying the values for these error matrices is not an easy problem. In [22], the authors use a combination of sensor specifications and approximations to determine the appropriate values for the error covariance matrices. The authors of [46] use a large number of test runs, and use values for the covariance matrix parameters that minimize the difference between the actual and predicted values.

In order to determine the appropriate values for q_k and r_k under the friction model, we use a combination of the techniques used in [22, 46]. Using operator generated mouse profiles, which are simply time-indexed records of mouse positions generated by operators in the course of using the system for different tasks, such as the object location task described in Section 6.2.1, we can determine an empirical value for both q_k and r_k that minimizes the average mean square error over all the profiles. The values $q_k = 6.59$ pixels and $r_k = 3.29$ pixels are found to minimize this error. For each profile, the best value for q_k was computed using the difference between the actual values for $\mathbf{x}_k$, and the values calculated using the model. The value for r_k is calculated using the maximum possible measurement error, which for mouse measurements is assumed to be at most a single pixel due to quantization error, coupled with a fraction of q_k based on the measurement delay and prediction length.

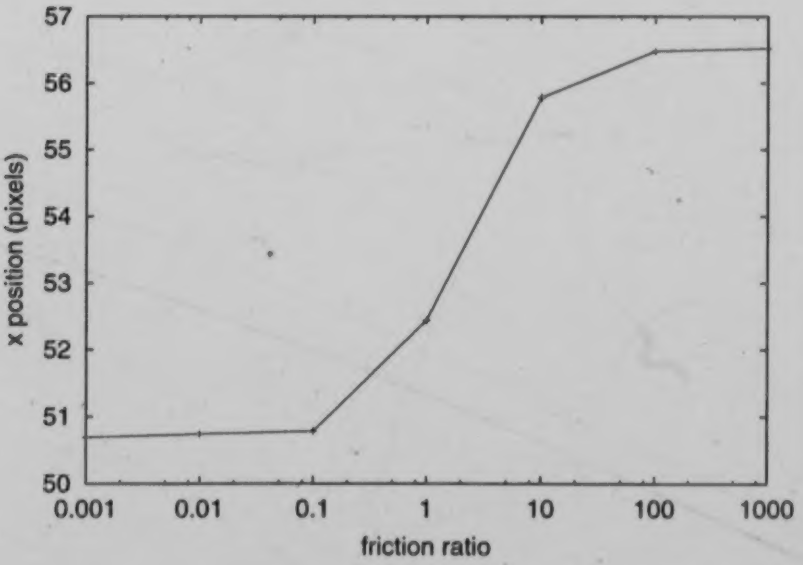
6.1.3 Prediction Evaluation

The first step in evaluating our predictive window technique is to evaluate the prediction process itself. In order to do this, we need to be able to reproduce a user's input and use it with the different prediction models. We record the sequence of mouse events that occurs during use of the telepresence system, and use the time of each event as an index. This sequence of mouse events is called a mouse profile. Several profiles are needed for an understanding of how accurately the predictor is performing. Each profile can be used with the different predictor models described in Section 6.1 above.

Our first measure of the accuracy of the prediction is a mean distance error calculation. The error is determined to be the Euclidean distance between the actual and predicted positions of the mouse, in pixels, at a given time (if there is no actual position, we interpolate from the positions before and after the requested time). The accuracy of the prediction can be determined from a simple set of mouse events.

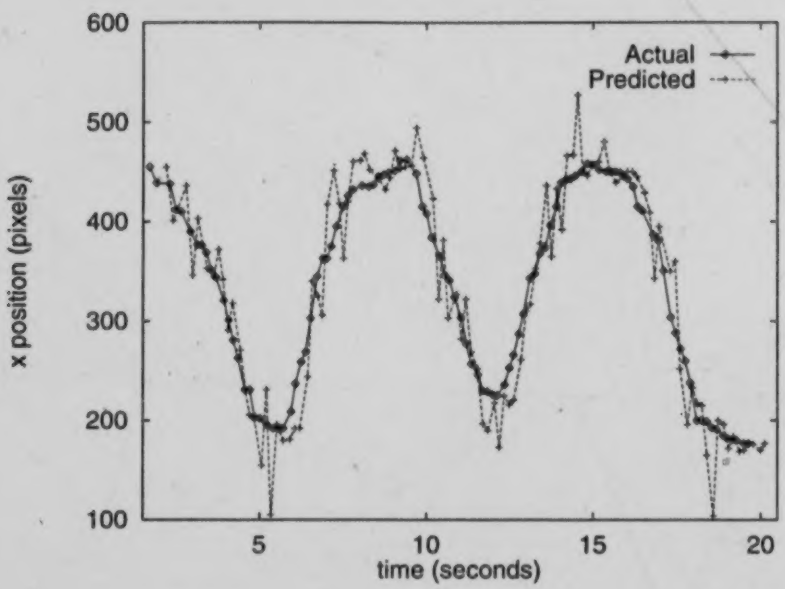


(a)

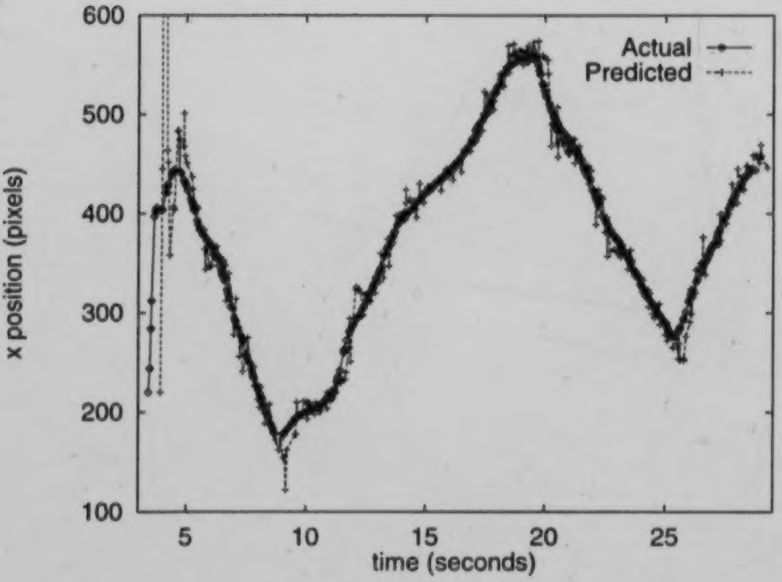


(b)

Figure 6.1: Mean distance error based on magnitude of the friction ratio for two different mouse motion sequences.



(a)



(b)

Figure 6.2: Predicted and actual values of x position in pixels for two mouse motion sequences. The prediction was done using the frictional model with a friction ratio of $f_r = 0.001$.

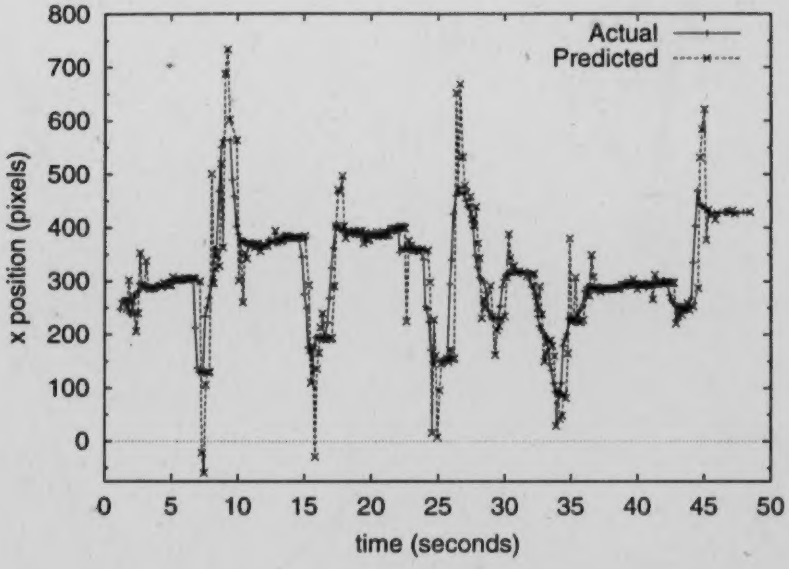
control profile	no pred.	const. vel.	const. accel.	friction $f_r = 0.001$
C1	64.22	45.29	47.65	50.69
C2	39.60	20.73	22.37	23.28
C3	36.37	33.25	35.60	39.79
C4	54.20	35.46	37.70	38.67
C5	37.49	25.80	26.57	30.04

Table 6.1: Mean distance error in pixels for each predictor model. Data is from five mouse profiles based on controlled mouse trajectories.

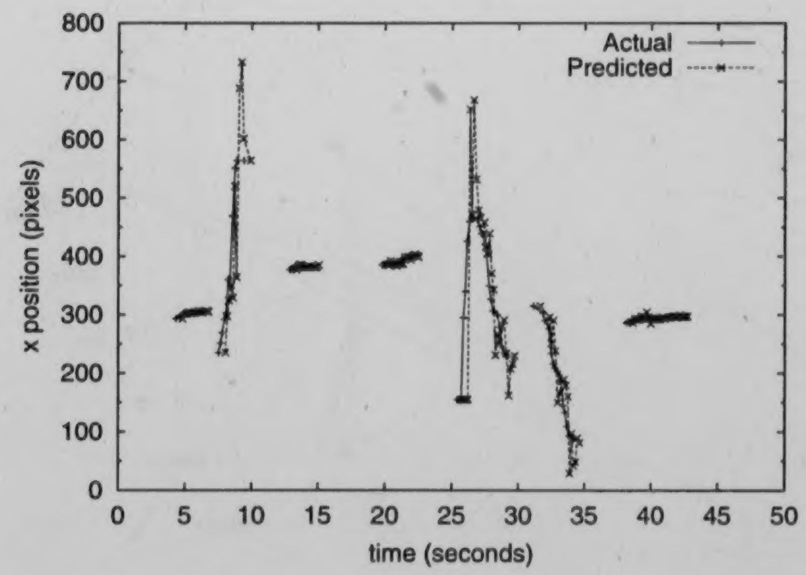
Comparison of Prediction Models

To determine the most accurate general model, we compare the different prediction models (constant velocity, constant acceleration, and friction) as well as using no prediction. Since the friction model includes the tunable parameter f_r that can change the accuracy of the prediction, we need to determine an optimal value for this parameter before the friction model can be compared to the other models. The graphs in Figure 6.1 show the mean distance error values associated with two different controlled mouse profiles as they relate to the order of magnitude of the friction ratio f_r. We can see that a ratio of less than 0.001 gives the best results. Figure 6.2 shows the accuracy of the predicted, one-dimensional mouse position, as determined by using the frictional model, compared to the actual position for two different mouse event sequences. These graphs show that when the mouse is being moved consistently, without sudden changes in direction or speed, the prediction is accurate. When the mouse suddenly changes direction, or suddenly stops, the predictor requires a few more events to recover and return to the same level of accuracy. The initial errors in the graph of Figure 6.2 (b) are due to the initial rapid motion followed by much slower motion as we can see in the "Actual" curve.

The graph in Figure 6.3 (a) shows the predicted versus actual positions using the velocity predictor on a full mouse profile. We see the same sort of results as in Figure 6.2 – accurate prediction except when sudden changes in motion occur. Figure 6.3 (b) shows a graph of the predicted mouse position versus the actual mouse position, when only the mouse events corresponding



(a)



(b)

Figure 6.3: Prediction of x position (vertical axis) in pixels using constant velocity model vs. time (horizontal axis) in seconds.

user profile	no pred.	const. vel.	const. accel.	friction $f_r = 0.001$
U1	67.250	57.119	58.056	64.018
U2	56.430	57.441	60.760	66.255
U3	47.497	45.905	46.717	53.508
U4	40.608	38.294	39.678	43.619
U5	50.802	49.701	51.339	55.553

Table 6.2: Mean distance error in pixels for each predictor model. Data is from five mouse profiles based on user controlled mouse trajectories.

to moving the image frame are used for the prediction. This graph emphasizes the accuracy of the predictor when there are no sudden changes in acceleration in the mouse motion. The breaks in the graphed values are when the operator is not moving the image frame. He is either repositioning the mouse to move the frame again, or suspending movement of the mouse.

Table 6.1 shows the results when control profiles are used to compare predictors. The mouse profiles for this table were generated using deliberately controlled mouse motion with smooth changes in the direction of motion. In this table, we see that using some predictor is seems to consistently provide more accurate results than using no predictor, and that the constant velocity model produces the best results. This shows that operators exhibit behaviours that can be predicted.

Table 6.2 compares the different predictors with using no prediction on several mouse profiles. These profiles are from operators performing the object location task described in Section 6.2.1. Since we are interested in the prediction when the small image frame is actually being moved, we disregard mouse events that are simply repositioning of the mouse before performing the move. With this technique, there is the possibility of registering no mouse events over an extended period of time. When this occurs, we reset the predictor once we receive a new mouse event. We can see from the table that using a predictor gives a better result in four of the five profiles than using no predictor, and that the constant velocity model gives the best results among all the prediction models.

Figure 6.4 shows the x position of an actual operator profile along with the

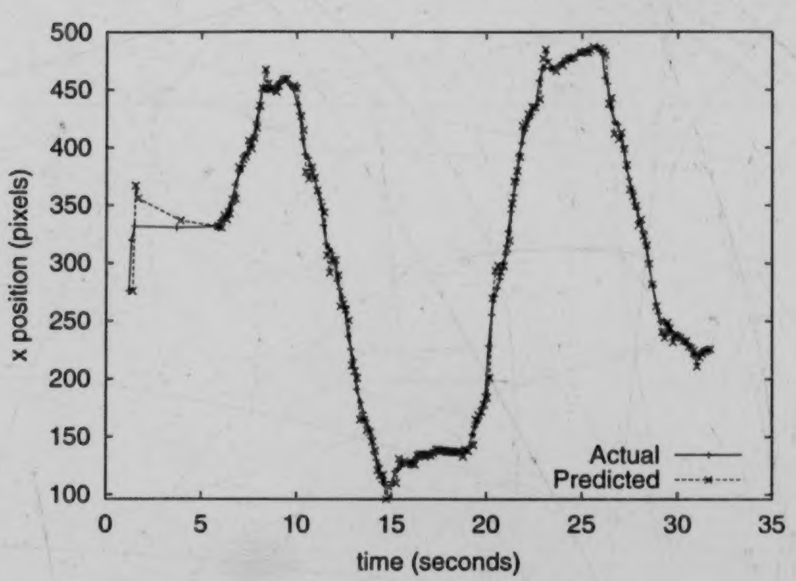


Figure 6.4: Predicted and actual values of x position, in pixels. Shows relative smoothness of predicted path over gentle motion and stabilization after sudden changes in motion.

profile	no pred.	old model	error model	% of original
U6	14.857	10.238	9.343	62.9%
U7	10.385	5.170	4.878	47.0%
U8	13.043	11.204	11.688	90.0%
U9	9.250	6.549	6.043	65.3%
U10	0.817	1.310	1.042	127.5%

Table 6.3: Mean distance error in pixels derived using covariance modeling. Data is from five mouse profiles based on operator's mouse trajectories.

predicted value for the next time step. We can see in the graph that the prediction works quite well when the mouse has a non-jerky motion. When there are sudden changes in direction, the model takes a few samples to stabilize and return to a close approximation of the trajectory.

The results presented in Tables 6.1 and 6.2 show the prediction when the error covariance matrices $\mathbf{Q}_k$ and $\mathbf{R}_k$ were set to the identity matrix. In Section 6.1.2, the final values for these error matrices are presented. Table 6.3 shows a comparison between results with and without covariance modeling. We can see that the predictive friction model generally produces a smaller mean distance error than no prediction, and that the covariance modeling improves this still further. Since the covariance modeling was based on averaging over actual mouse profiles, there are some cases where the new prediction model does not achieve a better result.

6.2 Experiments

The previous section presented the details behind the predictive Kalman filter approach used for predicting the motion of viewing windows. In this section, experimental results verifying the effectiveness of this approach are presented, applying it to two different types of tasks that are performed in mobile telerobotic navigation applications. These tasks, object location and object tracking, form the basis of telenavigation. Whenever an operator is trying to navigate in an environment, they must determine where they are going, i.e. object location, and then move to that location while generally keeping the location in view, i.e. object tracking.

Section 6.1.3 has shown that, from a purely numerical point of view, using prediction appears to provide more accurate results, in general, than not using prediction. In this section, several experiments are presented to show that using prediction on viewing direction provides an operator with a user interface for telerobotic navigation tasks that appears to have improved video quality. The experiments involve the operator performing several tasks while evaluating the video quality of the interface under the different prediction models. These tasks include: object location using flat, limited field–of–view images; a similar task using panoramic images; and an object following task using panoramic images.

6.2.1 Object Location Using Limited Field-Of-View Images

Initially, our predictive window experiments were conducted using a standard 640x480 limited field-of-view video stream. These experiments were a simu-

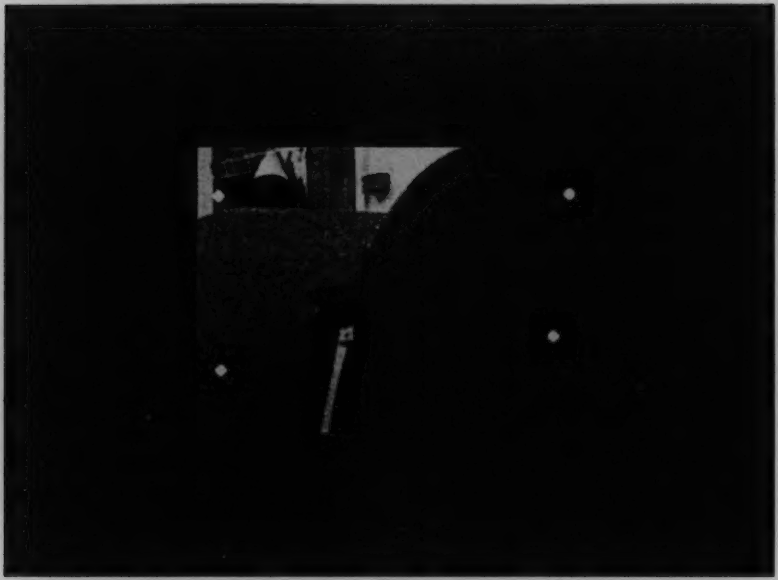


Figure 6.5: User interface for object location experiment. The window is 640 × 480 pixels in size while the inner image has 320 × 240 pixels.

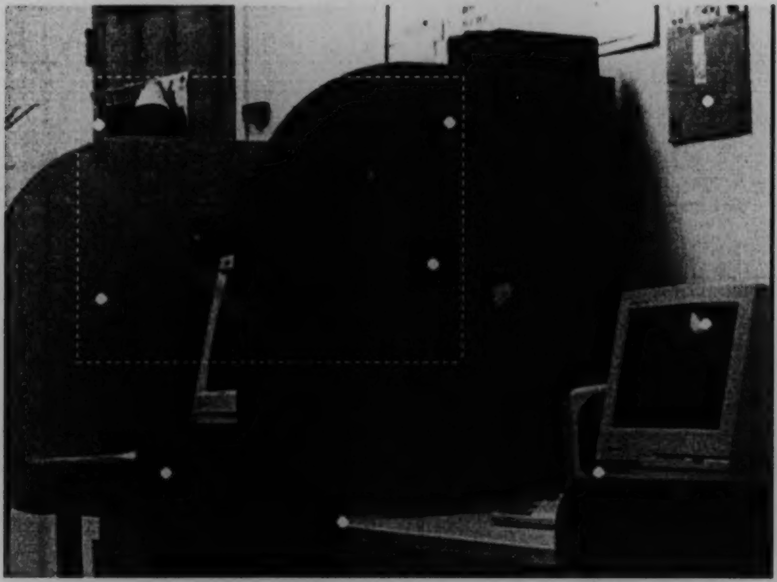


Figure 6.6: Full image from object location experiment. The broken lines outline the viewing window that the operator is currently looking at in the central portion of the user interface (Figure 6.5).

lation of an object location task using a static camera. We chose this setup as our initial evaluation experiment since predicting operator motion is not dependent on camera motion, and having a scene change too much at any given time might become a distraction for the operator.

In the initial task, a group of nine circles, each with a radius of 5 pixels, were graphically inserted into the full video stream, 640 × 480 pixels, in a perturbed three by three grid. The image is divided into nine sections of 213 × 160 pixels. Each section contains a single circle randomly placed within the section. The circle can be located anywhere between 53 and 159 pixels from the left side and between 40 and 120 pixels from the top of the section. This random placement ensures that no more than four circles can be viewed at any time.

The grid is perturbed between experiments so that the operator does not become trained to look in exactly the same location on each test. A grid base is used so that the circles are not concentrated in one place and the operator needs to move the viewing window in order to locate all the objects. Figure 6.5 shows a screen capture of the user interface as seen by an operator during this experiment. Figure 6.6 shows what the full image would look like with all of the circles shown.

When the experiment is started, the display window, 320×240 pixels, shows the centre of the image. The operator controls the viewing window using a "click-and-drag" motion. While the operator holds down the left button and moves the mouse, the viewing window follows the motion of the mouse. The operator's task is to locate and select all of the circles. Each circle is selected by placing the mouse pointer over the circle and clicking the right mouse button. When the pointer is within the 10 × 10 square containing the circle and the button is pressed, the circle is removed from the image and deemed to have been located. Since not all of the circles can be in the viewing window at the same time, the operator has to search most of the image by moving the viewing window until all the circles are located and selected.

This experiment seeks feedback from the operators on which prediction method was preferred; "no prediction" was also a permissible choice. Each operator was asked to perform the location task at least three times using each prediction model, as described in Section 6.1, and also without prediction. The models are originally presented in a random order. The operators are allowed to use each model as many times as needed to decide which they prefer. Once an operator has performed the task with all prediction models, they are asked which predictor they prefer and to justify their choice.

Five test subjects, all Computing Science graduate students or staff, are used for this phase of the experimentation. Of these subjects, two of them either had no preferred predictor or showed no strong preference for any given predictor. The remaining three preferred the full friction model and the constant velocity model equally. The experiments do not evaluate how quickly an operator could perform this task, only perceived video quality.

A secondary criterion concerns quality when only partial data is being transmitted, i.e., what to display when there is no new data available for a given portion of the new viewing window. Each of the three experiments, for each predictor, used a slightly different display technique when there was no image data transmitted for a given section of the image. These methods were:

1. display old data if the operator moves the image frame beyond the bounds of the current frame;
2. display nothing if the operator moves the image frame beyond the bounds; and
3. do not allow the operator to move the image frame beyond the bounds of the current frame.

When asked which of these display methods was preferred, three of the subjects preferred the first method (displaying old data), one subject preferred the first and second methods (display old data and display nothing) equally, while the last subject preferred the third method (no motion of the image into sections with only old data). This raises the interesting question of how the display method affects which predictor is preferred by the operator.

6.2.2 Object Location Using Panoramic Video

Applications better suited for predictive viewing windows involve the use of a panoramic video system. It is possible to transmit the entire panoramic image for every frame update, but that would require increased bandwidth. By predicting viewing direction, we transmit only the portion of the image the operator should be viewing during the next few time intervals. To prove the validity of predicting viewing direction with a panoramic imaging system, two simple tasks that can be done using such a system were selected. For a mobile telerobotic application, locating either objects or goals by the operator can be an important first step in completing a navigation task, so the first experiment involves another object location task.

The panoramic object location task asks each operator to find, as quickly as possible, five shapes arranged around the camera. The display shows the operator a 60 degree subsection of the full panoramic image in a 320 × 240 pixel window. The mouse is used in the "click-and-drag" approach (described in Section 6.2.1) to move the viewing window in the panoramic image. The five shapes (a rectangle, a circle, a pentagon, a triangle, and an irregular shape) are coloured black and attached to a white cylinder which is placed around the camera post. The shapes are placed so that they can be viewed in a central band in the viewing window. Multiple placements are made and can be changed between experiments to reduce the chance that an operator will memorize the placement of the shapes making the shape location easier. For each experiment, the placement set and the order of selection are determined randomly.

The operator is told which shape to find first and, as each shape is found, is given the next shape. To identify a shape, the operator moves the mouse pointer over the shape and pressed the right mouse button. This places a while circle on the image at the location of the selection. The operator's choice is watched by a human observer to ensure that the correct shape has been selected accurately.

The operator is asked to perform the task several times to evaluate two prediction modes and determine which was preferred. The two modes being tested are: prediction using the friction model, and no prediction. The task can be considered a simulation since there is no robot motion involved. However, location of an object before moving to it is an important part of the navigation process.

This experiment was conducted with a group of six operators. All the operators were computing science graduate students or professors. Initially, each operator was asked to perform the task as quickly as possible, and decide if using prediction provided a better interface than not using any prediction. Neither time to complete the task, nor preference, seemed to be determined by prediction or the lack of it. Even after spending time training on the interface, the first attempt at the task always took longer. In contrast, when asked to complete the task quickly, but pay attention to the video quality, the consensus was that the quality was better when prediction was being used. Of the six operators, five of them preferred the video quality when prediction was being used.

6.2.3 Object Tracking Using Panoramic Video

The object tracking task is an experiment in which the operator is shown a segment of a static panoramic image where a graphical object – a ball with a diameter of 20 pixels – is injected into the scene (see Figure 6.7). The scene shows a 60 degree section of the full panoramic image in a 320 × 240 pixel viewing window. A 40 × 40 pixel square outline is also placed in the viewing window. This is a targeting square to show the operator where the ball should be contained.

The ball is moved around in the image as if under the influence of a simple equation of motion that includes a occasional random acceleration component. The basic equation of motion of the ball can be written as the recursive state equations:

$$\begin{aligned}\theta_{k+1} &= \theta_k + v_k \Delta t + \frac{1}{2}(a_k + \Delta a)\Delta t^2 \\ v_{k+1} &= v_k + (a_k + \Delta a)\Delta t\end{aligned}$$

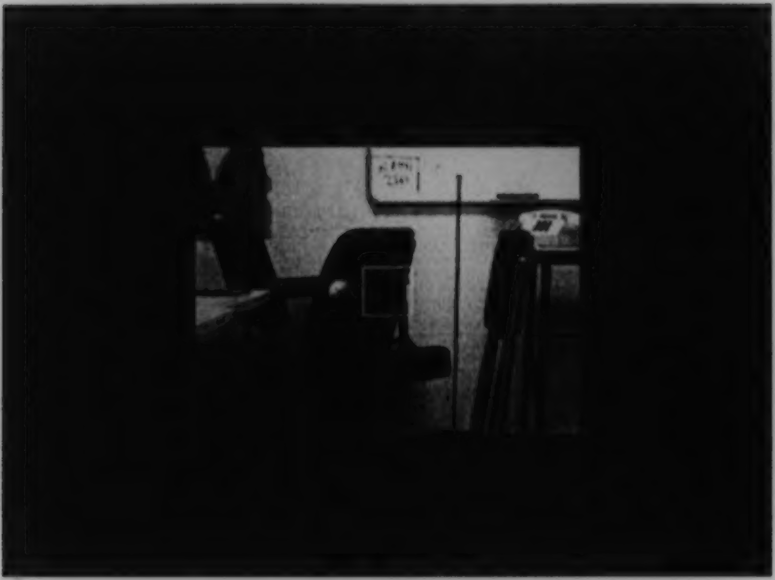


Figure 6.7: Sample of the tracking experiment interface. The white ball is the object being tracked by the operator, while the central square indicates the centre of the image. The operator is trying to keep the ball contained within the square as much as possible. Original image resolution is 640 × 480. The central image is a 320×240 portion of the full image shown in Figures 6.9 and 6.10.

$$a_{k+1} = a_k + \Delta a$$

where θ_k is the angular position of the centre of the ball with respect to the centre of the panoramic image (the radial position is constant and places the ball in a central strip of the viewing image), v_k is the angular velocity, a_k is the angular acceleration, and Δa is the random acceleration component.

The value of Δa is normally 0, but at randomly determined time steps, initially the tenth step, then at random intervals between zero and 24 time steps, the value of Δa is set to a random value:

$$\Delta a = -drand48()\frac{\pi}{4}\frac{t}{30}\frac{v_k}{|v_k|}$$

where $drand48()$ is a C random number generator producing a value in the range $[0, 1)$, t is the current time, and v_k is the current velocity. This value of

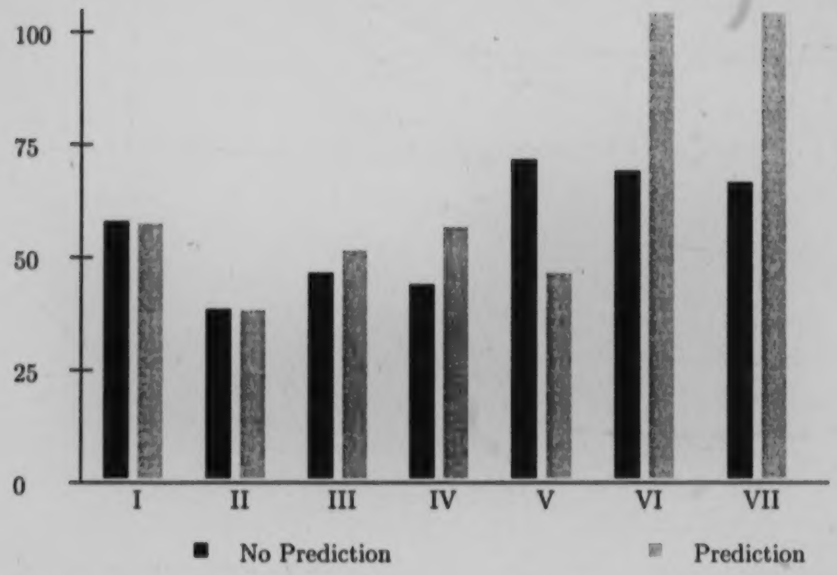


Figure 6.8: Summary of tracking times (in seconds along the vertical axis) for seven different operators in panoramic tracking experiment. Each operator performed with prediction (frictional model with covariance modeling) and without prediction.

Δa is used in the current calculations and then reset to 0 for the next time step.

The operator is asked to follow the ball and keep it as close to the centre of the viewing window as possible, within the outline of the square. As time progresses, the magnitude of the random acceleration increases so that the ball becomes harder to track. The objective of the experiment is to track the ball for as long as possible. Since the ball becomes more erratic as time passes, we wish to determine if using prediction allows the operator to track the ball for a longer period of time.

Each operator was asked to perform the task using both prediction using the friction model, and no prediction. The amount of time each operator spent tracking the ball under the two predictors is summarized in the graph in Figure 6.8 (the vertical axis shows the length of time for each experiment). We can see that four of the seven operators – numbers III, IV, VI, and VII – had markedly better performance when prediction was being used; while

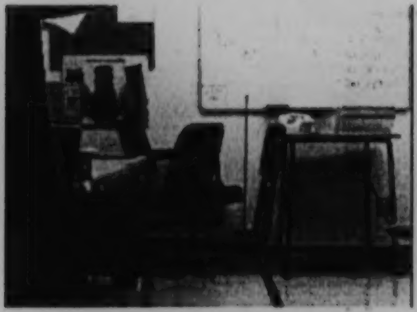


Figure 6.9: Image showing blackness for mouse profile, time 18.5 seconds, when prediction is being used. The image resolution is 640 × 480 reduced to 320 × 240.

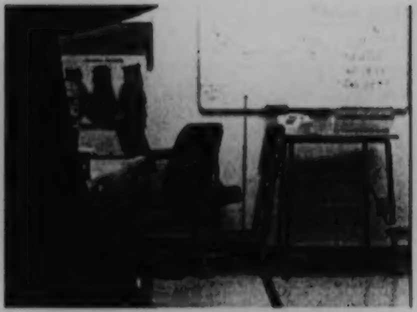


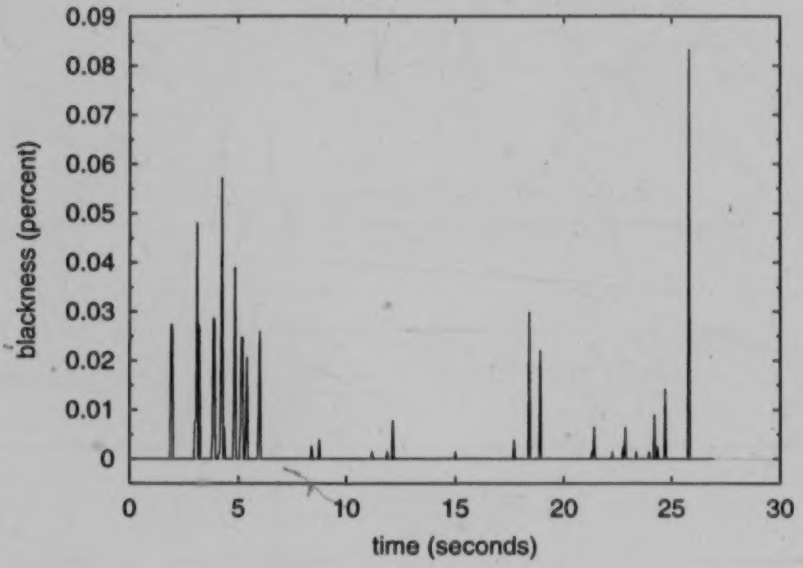
Figure 6.10: Image showing blackness for mouse profile, time 18.5 seconds, when no prediction is being used. The image resolution is 640 × 480 reduced to 320 × 240.

operator V performed better when there was no prediction. The remaining two operators (I and II) had inconclusive results, where the times were very similar. When asked, all the operators stated that, in their opinion, using prediction generated better quality images than experiments that used no prediction.

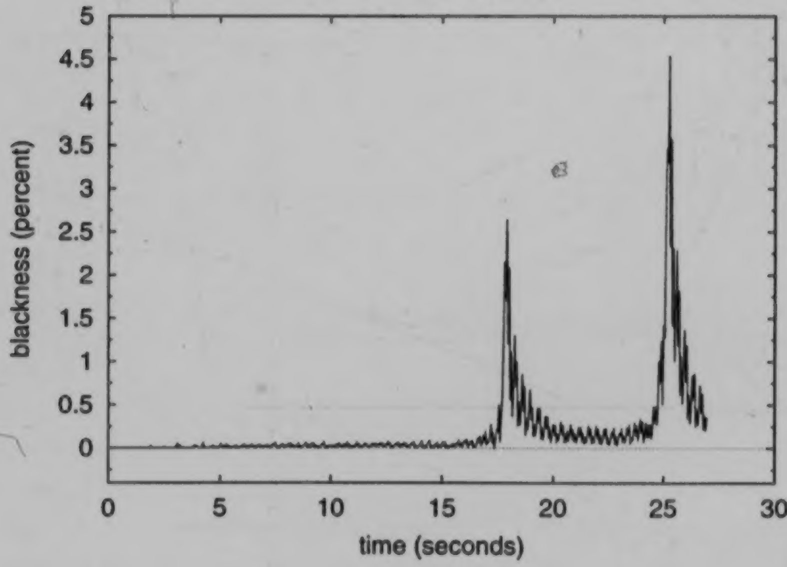
6.2.4 Measuring Image Quality

To measure the quality of an image, I_i, we propose measuring the amount of image I_i, which is being shown to the operator, that has not been transmitted to the user interface. This measure is called the *blackness* of the image, $B(I_i)$ because the sections of the image that have no data are displayed as black areas. Figures 6.9 and 6.10 show images that contain blackness areas when portions of the data have not been transmitted. The blackness always starts from the corners of the image because a rectangular sub-image containing the requested wedge is always transmitted. This approach can transmit more data than requested, and this leads to a curved black area appearing at the corners of one side of the image.

The amount of blackness, $B(I_i)$, in an image frame I_i is measured by counting the number of black, intensity value 0, pixels. In a normal image, there are very few pixels that will naturally be black (in Figures 6.9 and 6.10 there are no truly black, intensity 0, pixels). This means that any pixels that



(a) Prediction



(b) No prediction

Figure 6.11: Percentage of blackness for each frame during a controlled trajectory; shows progression of blackness for trajectory under both prediction and no prediction cases. The trajectory is a constant acceleration trajectory.

have a value of 0 can be assumed to not contain any real information. The blackness for frame I_i is:

$$B(I_i) = \sum_{j=1}^{W} \sum_{k=1}^{L} \neg I_i(j,k)$$

where image I_i has width W and height L, and for pixel $I_i(j,k)$

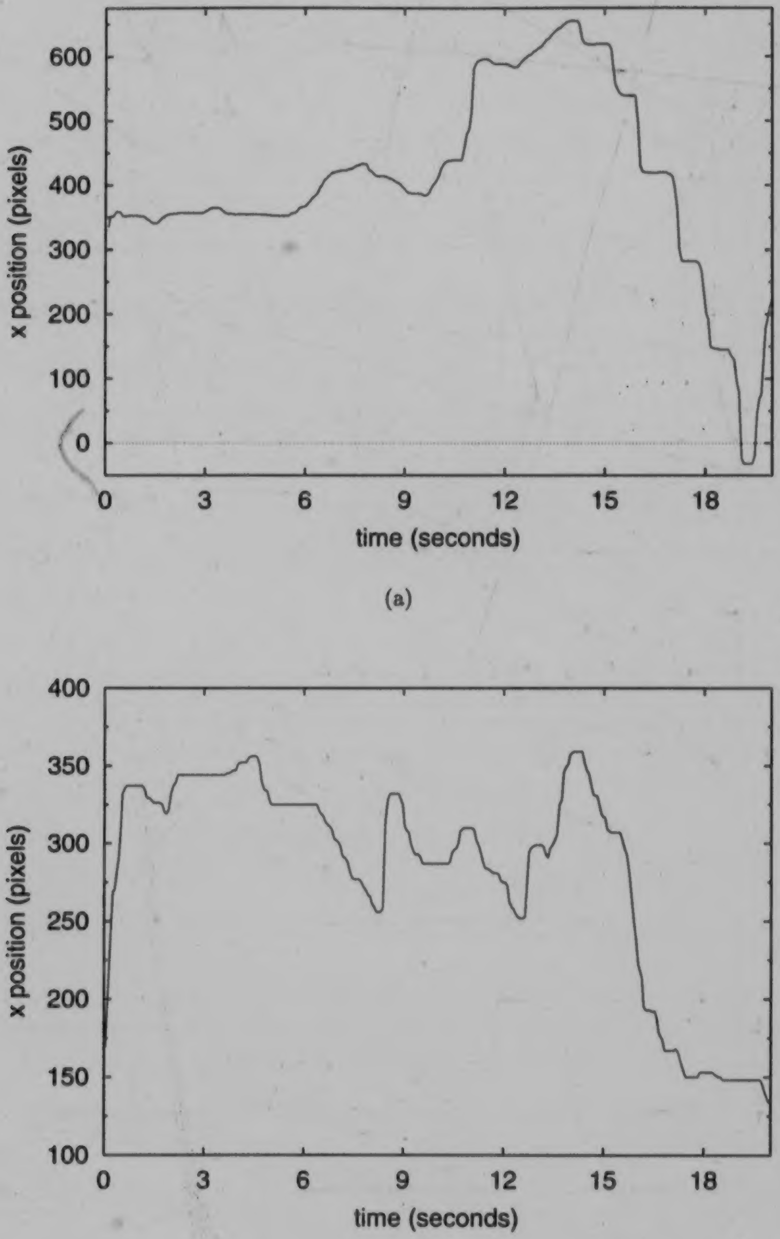
$$\neg I_i(j,k) = \begin{cases} 1 & \text{if } I_i(j,k) = 0 \\ 0 & \text{otherwise} \end{cases}$$

The overall quality for the image sequence is calculated using the average blackness for all images in the sequence of length N, $\bar{B} = \frac{1}{N} \sum_{i=1}^{N} B(I_i)$. This allows a comparison of the quality of different image sequences. If the percentage of black pixels is used, by dividing $\bar{B}$ for a given sequence by the size of the displayed image $W \times L$, the quality for image sequences displaying different size images can be compared.

With an image quality measure, the results from prediction and no prediction experiments can be compared. There are two types of data to use in this comparison: data from controlled trajectories and data from operator experiments. Controlled trajectories are experiments where the object being tracked assumes a repeatable path and the tracking is also on a repeatable path. These types of trajectories include constant velocity trajectories and constant acceleration trajectories.

Figure 6.11 shows a graph of the percentage of black pixels in each image frame, at 20 frames per second, for a constant acceleration trajectory over 50 seconds. The percentage of blackness when prediction is used is barely visible along the bottom edge of the graph and much lower than when no prediction is used, with $\bar{B}$ equal to 0.274 and 61.068 respectively. Similar results can be shown for a constant velocity trajectory. This figure demonstrates that with controlled trajectories, using prediction gives much better apparent image quality than no prediction as measured by the blackness.

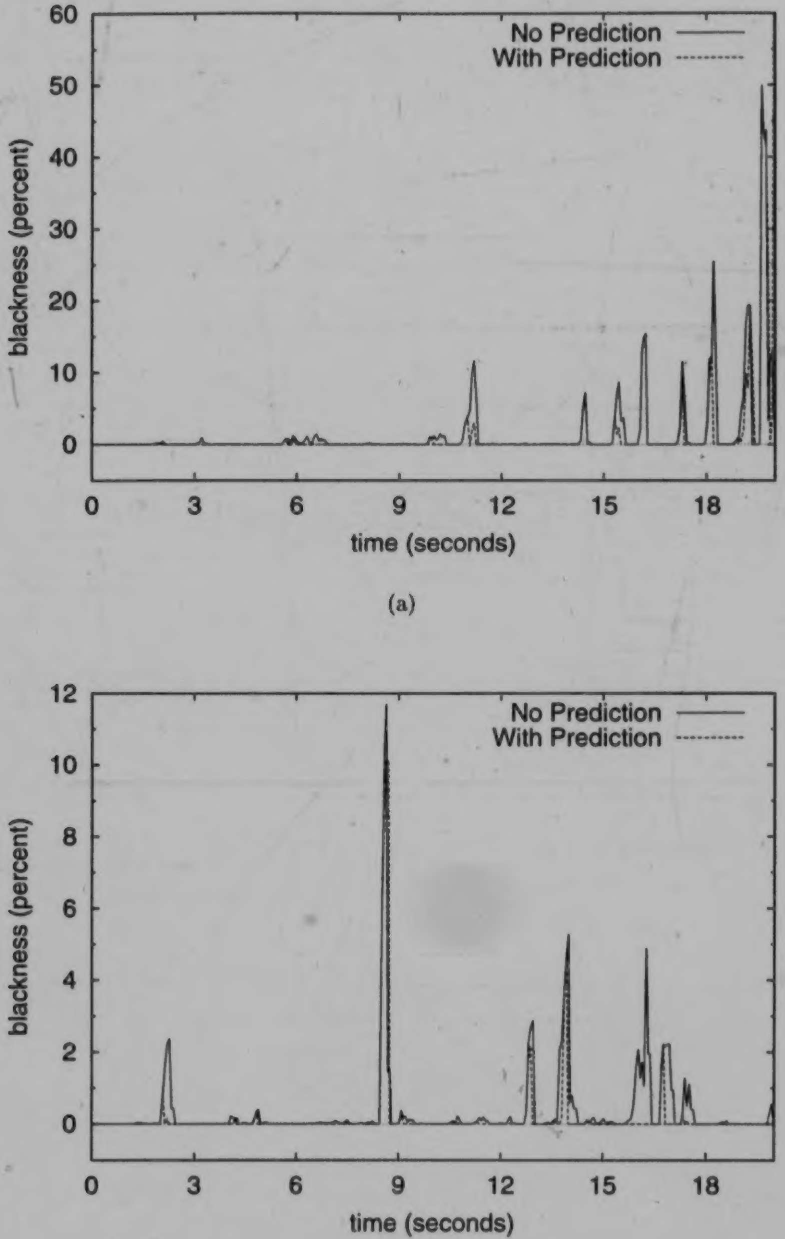
To make image quality comparisons based on user data, stored ball trajectories are used. For these trajectories, the operator is assumed to be capable of perfect tracking, and prediction is made based on the ball motion. Comparison between the quality of the resulting image sequences can now be made.



(a)

(b)

Figure 6.12: First 20 seconds of x coordinate of mouse positions, in pixels, for two different trajectories, (a) and (b), during panoramic tracking experiment.



(a)

(b)

Figure 6.13: Percentage of blackness for each frame for the first 20 seconds of Trajectories (a) and (b) from Figure 6.12. Shows percentage for both prediction and no prediction cases.

	prediction	no prediction
Trajectory 1	1.43%	1.47%
Trajectory 2	0.17%	0.33%
Trajectory 3	0.11%	0.33%
Trajectory 4	0.20%	0.49%
Trajectory 5	0.12%	0.23%

Table 6.4: Blackness calculation for first 20 seconds of 5 trajectories showing average blackness for both prediction and no prediction.

Figure 6.12 shows the first 20 seconds of two different ball motion profiles. Each one shows the x position over time for two different operators. The average amount of blackness for the first 20 seconds of these two trajectories – trajectories 1 and 2 respectively – and three other trajectories, is summarized in Table 6.4 for prediction using the friction model, and no prediction.

Table 6.4 shows that using prediction produces a smaller average blackness measure for each trajectory. This represents a video stream of better quality overall when prediction is used. Figure 6.13 shows the percentage of blackness from trajectories 1 and 2 again over the first 20 seconds for prediction and no prediction. In general, using prediction produces less blackness, but there are peaks where prediction produces more black than no prediction for the same time point. This usually occurs in sections where either the predictor is not performing well due to a sudden change in motion, or where the mouse motion changes too frequently between frame transmissions – for example, due to the different mouse polling and frame transmission rates.

6.3 Chapter Summary

This chapter has presented research done using mouse models and prediction to reduce the size of a viewed image from limited field-of-view and panoramic cameras. Prediction appears to reduce the delay seen by the operator when changing viewing directions. Experiments have been described that demonstrate the effectiveness of the prediction and the improvement of the operator's experience.

Chapter 7 will describe another approach to reducing the amount of bandwidth required for telenavigation applications. The approach presented, using feature images, can be used in combination with or as an alternative to traditional compression techniques. These feature images can include edges, segments, or motion. Each has advantages and drawbacks which will be discussed in more detail.

Chapter 7

Low Bandwidth Feature Images

In the previous chapter, the bandwidth requirement for the video stream was reduced by using a process to predict the motion of the mouse used to control a viewing window. Using this smaller viewing window decreases the amount of bandwidth required by reducing the size of the image being transmitted. While this leads to a small amount of reduction (based on the size of the viewing window), for very low bandwidth applications, such as teleoperation over the Internet, a larger reduction is needed.

The traditional approach at this point is to use compression techniques such as MPEG, H.261, or H.263 depending on the available bandwidth. Unfortunately, all of these approaches require a constant size window, which is not available if mouse prediction is being used. Each of these techniques have other drawbacks which may make using them undesirable.

For MPEG-1 and MPEG-2, the amount of bandwidth required is quite high. Both of these compression standards were designed for bit rates of greater than 1 Mbps. MPEG-1 can be used with much lower bandwidth, but with a significant degradation in video quality. While neither is ideal for low bandwidth applications, they could be used if necessary.

The H.26x compression standards would be more appropriate for low bandwidth applications, but also have drawbacks. Both standards assume a scene composed of very limited motion, such as the talking head. In a telenavigation application, the whole scene will be moving and sometimes quite quickly. With large amounts of motion, a significant reduction in quality is expected from the H.26x standards.

What is being proposed in this chapter is to transmit feature images in very low bandwidth applications. These feature images can contain a very small amount of data, depending on the feature, because only the individual features are being transmitted not necessarily all the texture and shading information. Generating these feature images can either be a replacement for traditional compression techniques or as a preprocessing step before applying other compression techniques. MPEG-4 may provide a standard that can be used to transmit feature images, however, the standard has only recently been finalized and was not used in this research.

One of the drawbacks to using feature images is that the application must be known in advance. Traditional compression can be applied in almost any application with some exceptions (H.261 and H.263 work best in teleconference applications, VR compression assumes a single unmoving point of interest, etc.). Feature images are much more task dependent. For example edge images will not work if the task is to find all of the blue balls in a group of balls, but segments might if the different coloured balls also have different colour intensities.

There are many different types of image features that can be extracted, such as edge, segment and motion features. Each of these image feature types can be extracted using existing image processing techniques. These techniques may not be optimal, but refining them is other research. Most feature extraction methods rely on a criterion of some form to determine whether an area of the image should be extracted as a feature. For example, the Sobel operator for edge detection compares the pixel mask with a threshold to determine if the pixel being checked is on an edge (see Section 7.2 for more details).

This chapter will discuss different types of feature images and their possible applications. Section 7.1 will discuss bandwidth requirements for video in very low bandwidth applications. In Section 7.2, edge features will be discussed. Finally, Sections 7.3 and 7.4 will describe segment and motion features.

7.1 Network Considerations

When there is only very low available bandwidth, such as using phone lines or congested Internet connections, the amount of data being transmitted needs to be reduced. There are two aspects that need to be considered when reducing the amount of data. The first is the final amount of data that will be reduced. The second consideration is the time required to reduce the data.

In this research, the minimum amount of bandwidth available is the presumed to be that available using an analog telephone line. Section 2.6 has shown that this technology guarantees to transmit 25,000 bps. As technology improves, this transmission capacity will certainly increase, but this appears to be the average bandwidth provided over the Internet during business hours. Although the total available bandwidth of the Internet will increase with technology improvements, the number of people using the network will also increase. This means that the bandwidth available to an individual will probably not increase significantly.

For the remainder of this chapter, the minimum amount of bandwidth available for the program will be 25,000 bps. Using this figure, the minimum frame size and delay figures can be calculated.

Although a video stream should provide 25 to 30 frames per second, when only very low bandwidth is available, this requires a frame to be extremely small. To provide 30 frames per second, each frame can be no larger than 833 bits or approximately 100 bytes. If the original frame size is 320 × 240 with 8 bit per pixel grey-scale, which is the display size used in the teleoperation interface, compression and other reductions must reduce the amount of data to be transmitted by a factor of 768. This is a very large compression factor.

A frame size of 200 bytes, still requiring a large compression factor, still provides 15 frames per second. This provides a video stream that is not maintained at ideal rates, but allows the operator to control the telerobot accurately. In the remainder of the discussion, a frame size of 200 bytes would be considered very good for this available bandwidth.

If a full speed frame rate, 30 frames per second, is desired, a single frame must take 33 ms to transmit. At 15 frames per second, as discussed above, a single frame will take 66 ms to transmit. This provides a measure for the maximum processing time for a single frame. During the transmission of the current frame, any processing for the next frame can be completed. When the current frame has finished being transmitted, the next frame must be ready. This will keep the lag between frames to a minimum.

7.2 Edge Features

In grey-scale image processing, an edge is a boundary between two regions with distinctly different grey-levels. These features can be extracted from an image using a variety of techniques including gradient operators or a Laplacian operator which is a second-derivative operator. It has been observed that edges, or light-dark contours, are probably the most important component of what we perceive [36].

In a navigation task, the colour and texture information is less important for the task than the edges of the environment. For example, in order to move down the hallway, knowing the location of the wall is more important than the contents of the posters on it. Object identification requires more information if a distinct object is required rather than any object of a given type.

Due to the important nature of edges, these are the primary type of feature image used in this research. From the research in teleprogramming and augmented reality, we can see that there is promise in the use of edge images for different types of tasks. Both of these research areas have used graphical wire-frame models overlayed on real-time images with good results. A graphical wire-frame model is very similar to an edge image and may even be derived from it.

There are many techniques that can be used for edge detection. The most appropriate techniques for this research need to generate edge images at 15 to 30 frames per second. A good summary and implementation of several commonly used edge detectors can be found in [55].

The simplest edge detectors convolve the original image with an image mask. This mask can be of any size: the **Roberts** operators use 2 × 2 masks while the **Prewitt** and **Sobel** operators both use 3 × 3 masks. Small masks can be processed quicker than large masks. The Sobel operators appear to produce reasonable edge images although the edges are thick (several pixels wide). A technique implementing these masks is fast and easy to implement; it will be discussed in more detail in Section 7.2.1.

The Sobel operator detects edge by approximating the gradient of the image. Another edge detection method involves using the Laplacian of the image:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

This is usually approximated by

$$\nabla^2 f = 4z_5 - (z_2 + z_4 + z_6 + z_8)$$

where z_5 is the pixel being processed by the mask and z_2, z_4, z_6, and z_8 are the pixels directly above, below, and to the sides. This approach is not commonly used by itself because it is susceptible to noise. It is usually used in combination with another technique.

Marr and Hildreth [49] introduced an edge detection method based on the Laplacian operator and the physiology of human vision. In their approach, the Laplacian is combined with a Gaussian mask to produce a new mask called the *Laplacian of the Gaussian*. Once the image has been convolved with this mask, the algorithm attempts to locate the zero crossings. It is these zero crossings that are the final results of the edge detection. The problem with this approach is that the masks can be quite large and processing time can be long. Gunn, in [33], examines in detail the relationship between the size of the mask and the predicted error in detecting edges.

Another method of detecting edges was presented in [16]. In this paper, Canny presents a computational approach to determining an optimal edge detector for step edges. The results of this computational approach are very similar to the method derived from the biological approach of Marr and Hildreth. In [57], Petrou and Kittler use a technique similar to that in [16] to derive an optimal edge detector for ramp edges. Unfortunately, both of these

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

Figure 7.1: Masks for Sobel operator calculation for the centre pixel of each mask (for G_x and G_y respectively)

methods, similar to the Laplacian of the Gaussian method, can be slow when compared to the Sobel operators.

7.2.1 Sobel Operator for Edge Detection

The gradient operator techniques are both accurate and fast. In particular, the Sobel operators [30] provide the necessary gradient operations with the added advantage of a smoothing effect. To determine an approximation to the gradient, the formula:

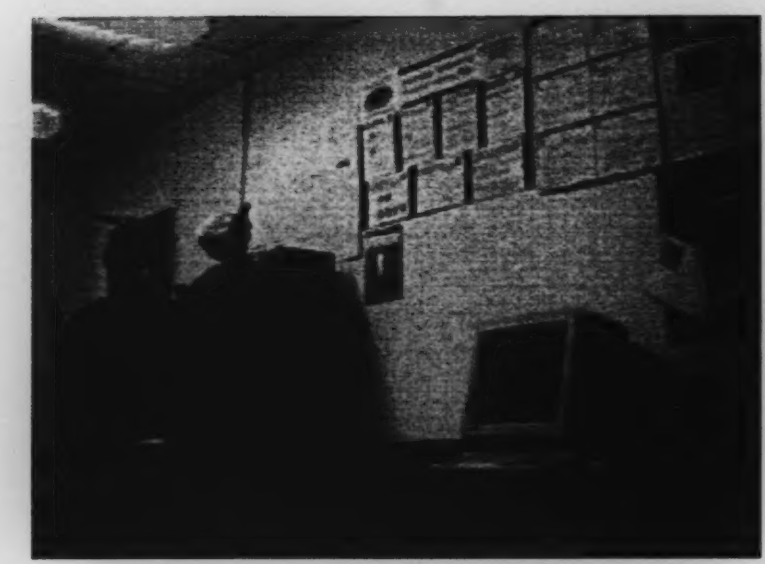
$$\nabla f \approx | G_x | + | G_y |$$

where $f(x, y)$ is the image and $G_x = \frac{\partial f}{\partial x}$, $G_y = \frac{\partial f}{\partial y}$ are the components of the gradient vector, can be used.

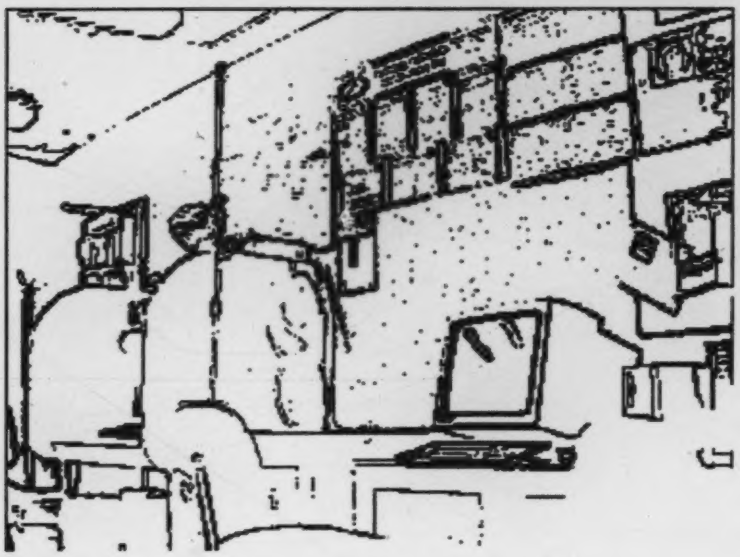
The Sobel masks are used (shown in Figure 7.1) to get values for G_x and G_y and thus a value for ∇f. To get an edge image using the Sobel operators, a threshold value for ∇f is also needed.

The major benefit in using the Sobel operators is that it is simple to implement and executes quickly. The algorithm applies the Sobel operator masks to each pixel in turn (with special treatment for the edges of the image) and compares the result to a threshold value in a single pass. Figure 7.2 shows an example of a grey-scale image and a generated edge image using this algorithm. The edge image was generated using a threshold value of 64.

The processing time for the edge detection algorithm using the Sobel operator depends on the size of the original image and not on its contents. This is because each pixel is processed in the same way, the mask is applied to it and its neighbours and is constant for the whole image. This implementation of the Sobel edge detector takes approximately 8.2 ms to process a single 320 × 240 image. This approximately $\frac{1}{4}$ of the time that can be allotted to processing



(a) Original image



(b) Generated edge image.

Figure 7.2: Sample edge image. Original image 640 × 480.

Threshold	64	75	100	128	150	175	200
Size (Bytes)	18,900	16,480	13,161	10,752	9,250	7,528	6,280
Comp. Ratio	16.25	18.64	23.34	28.57	33.21	40.81	48.92

Table 7.1: Compression ratio based on resulting size and edge threshold. The original image was 640 × 480 pixels or 307,200 bytes.

for 30 frames per second video (see Section 7.1) and hence is acceptable for processing the video stream.

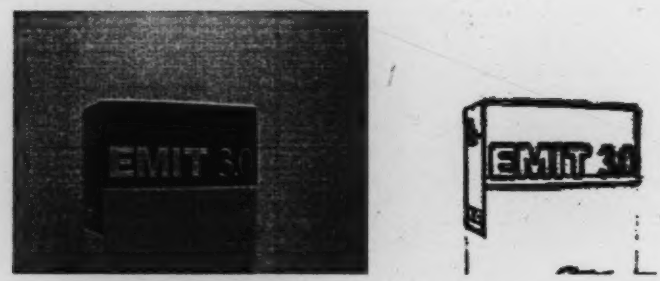
7.2.2 Compression of Edge Images

Since we are not interested in transmitting the whole image, we need to compress the resulting edge image. A simplistic approach is to use a run-length encoding scheme. In this algorithm, we simply count the number of pixels of a given grey-level, we can treat the image as a single array of pixels, and transmit the grey-level and the count for that value. For binary images, the grey-levels are either black or white. In this case, the count values alternate for each of these colours, so the grey-level does not have to be transmitted with the count.

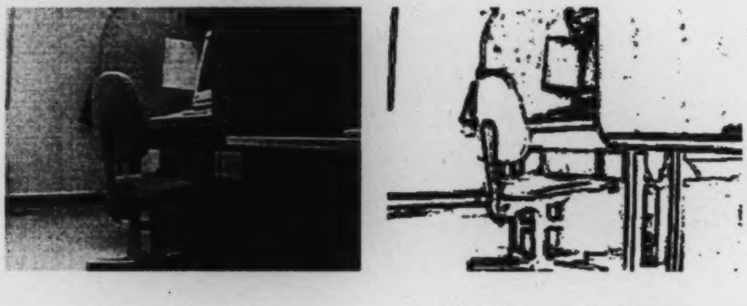
This approach is independent of the edge detection method applied. The only requirement is that the edge image is a binary image where a pixel is either part of an edge or not part of an edge. Provided the resulting edge image is not too noisy, this simple compression technique will provide a significant compression factor.

We can apply the run-length encoding algorithm to the example edge image in Figure 7.2. Table 7.1 shows the results of compressing images created using different thresholds. Using a threshold of 64, a compressed byte stream of 18,900 bytes is produced. The original image was 640×480 pixels and required 307,200 bytes. Using this simple run-length encoding compression scheme, a compression ratio of 16.25 is achieved. The edge image for this example still contains a large amount of noisy data. This noise can be reduced by increasing the threshold value used to generate the edge image. As shown in Table 7.1, increasing the threshold further increases the compression ratio.

Reducing the size of the original image produces a similar reduction in



(a) Partial box original. (b) Partial box edge image.



(c) Lab view original. (d) Lab view edge image.

Figure 7.3: Small images used for edge detection and run-length encoding. (a) and (c) are the originals, size 320 × 240 pixels. (b) and (d) show the results after edge detection with a threshold of 64 and before run-length encoding.

the compressed image. Figure 7.3 show two different images that have been processed using the Sobel edge detection algorithm and will be processed using the run-length encoding scheme.

The first produces a 1880 byte image after compression. Since the original image was 320 × 240 pixels, the process achieves a 40.85 compression ratio. This large compression is due mainly to the large, contiguous area of black near the top of the image. The compression takes approximately 1.6 ms giving a total processing time of approximately 10 ms.

The second image produces a compressed data stream which is 5952 bytes long. This results in a compression ratio of 12.9. This is not nearly as good as the previous image, but this image contains more noisy data. The compression takes approximately 1.8 ms to complete. This is still well within the acceptable processing time.

While the time to perform compression and decompression are both dependent on the input, both are completed very quickly. The decompression is slower to perform partly due to the different platform, the SGI Indy is much slower than the Pentium. The average time to accomplish the decompression is 2.9 ms but is still well within processing time tolerance.

7.2.3 Edge Lists

While using a run-length encoded compression technique reduces the amount of edge image data, there are other methods. One good candidate is to only transmit the end points of line segments. Unfortunately, this requires the system to determine line segments, but most edge detection algorithms, including the algorithms used here, only determine if a pixel is part of an edge.

If each line segment can be identified, a line can be transmitted by four numbers: the two end points. Using two bytes per number, each line segment can be identified by 8 bytes. This means that in 100 bytes, the smallest size for very low bandwidth applications, the 12 most significant edges of an image can be transmitted. In highly structured, indoor environments, this should be sufficient for simple navigation. Using 200 bytes per image allows 25 line segments.

Size Reduction	Process Time (s)
1	20-25
2	10.8
5	4.07

Table 7.2: Reduction in the size of searched Hough space vs. processing time.

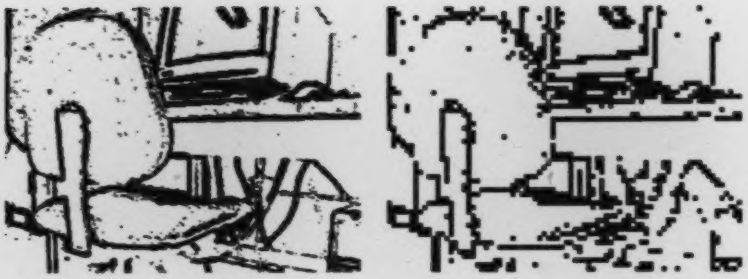
Line segments can be determined using the Hough transform [30]. Hough space can be used to annotate all possible lines parameterized by their distance to the origin and angle. The value for each element will be the number of pixels on that specific line.

There are several problems using this approach. The first is the time. Transforming an edge image into Hough space takes a significant amount of time. In order to experiment with this approach, a simple algorithm has been implemented. Unfortunately, this algorithm can take as long as 20-25 seconds to process a single 640 × 480 image.

The time to process an image can be reduced by using various approaches. The first is to increase the quantization size. By reducing the number of permissible angles and distances, the time to process can be reduced. Table 7.2 summarizes the processing times for reductions in the size Hough space. This table shows that reducing the size of the searched space does allow for a good reduction in the processing time, but it is still several orders of magnitude slower than the edge detection process. Reducing the permissible angles also produces a processing time reduction, but the errors created are much greater.

Another approach for reducing the amount of time the Hough transform takes is to use a subsampled edge image. Using the Sobel operators for detecting edges creates wide edges which allows for subsampling the image without losing too much information. Figure 7.4 shows a subsampled edge image beside the original. Other edge detectors, those that produce single pixel edges, may not survive the subsampling process without losing significant information. Averaging could be used instead of direct subsampling, but takes more time.

Table 7.3 shows the reductions in processing time as the Hough space



(a) Original edge image (b) Subsampled edge image.

Figure 7.4: Subsampled edge image. Original size of 320 × 240 pixels, subsampling every four pixels and every four lines.

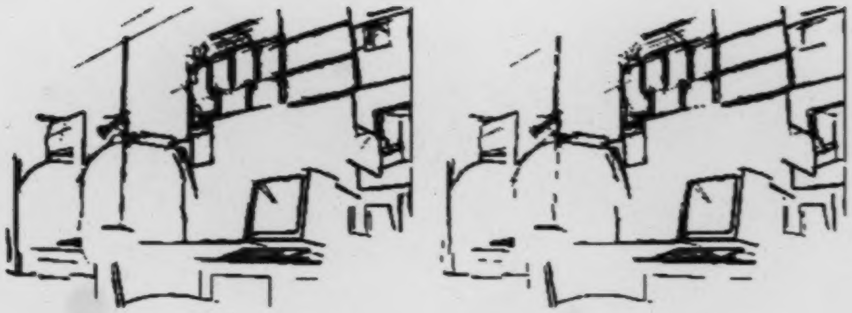
Subsample reduction	Hough reduction		
	1	2	5
2	5.45	2.63	1.03
2	2.66	1.27	0.49
2	1.71	0.67	0.28

Table 7.3: Processing time as a function of edge image subsampling and Hough space reduction.

reduction factor and subsampling factors are changed. In the table, the Hough space reduction factor is read across the top while the subsampling factor is shown down the left side. From the Table, it is apparent that reducing both factors reduces the processing time required, but the times are still significantly higher

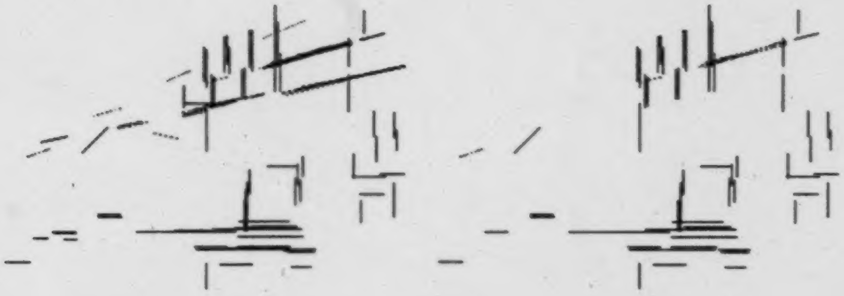
The last problem with generating edge lists is determining the significant edges. There are many criteria that could be used including length of edge (a threshold in Hough space), angle and location, number and size of gaps in the edge segment, etc. Once the criteria are established, each point in the Hough image is examined and, based on the previous criteria, generates potential end points. All of this takes time in addition to the edge detection.

While transforming the edge image into Hough space and extracting the line segments can allow significant reduction in the amount of the image data transmitted, it takes processing time. Reducing the processing time (using the methods described above) also reduces the accuracy of the transformation



(a) Full Hough space, no subsampling, 10,934 edges

(b) Hough space reduction 5, subsampling factor 2, 1,837edges



(c) Full Hough space, subsampling factor 4, 152 edges

(d) Hough space reduction 5, subsampling factor 4, 84 edges

Figure 7.5: Line images using the Hough transform with various parameters to reduce time required. The original image is from Figure 7.2.

and the perceived quality of the resulting images. Figure 7.5 shows images of varying quality generated using different sets of parameters to reduce the processing time. These figures are generated from the original and edge images shown in figure 7.2. As the time is reduced, the quality is obviously also reduced. As computers get faster, the processing time will be reduced by the technology, but at this time, using a Hough based approach is beyond the current computing technology under the real-time constraints assumed.

7.3 Segment Images

An image segment is a region in an image where all the pixels in that region have similar grey-level values. A segment image is an image where the segments have been identified. There are several methods that can be used to identify these segments. The first is to assign an arbitrary grey-level value to all the pixels in the region. This will provide an image where the segments can be easily identified, but will not guarantee representative grey-levels. A second method would be to use an average grey-level based on the original levels. This type of image would have grey-levels similar to the original, but may make it harder to distinguish between two regions if the threshold value for changing regions is small.

In telenavigation tasks, viewing only the segments may still allow the operator to perform adequately. Compression techniques using wavelets, such as [63, 69], show that images can still be recognized even at very low bit rates although the images produced can be quite blurry. Using segments can provide a similar type of image, one which appears blurred or blotchy, but the contents of the scene can be interpreted by a human operator although edges seem to be a more important feature.

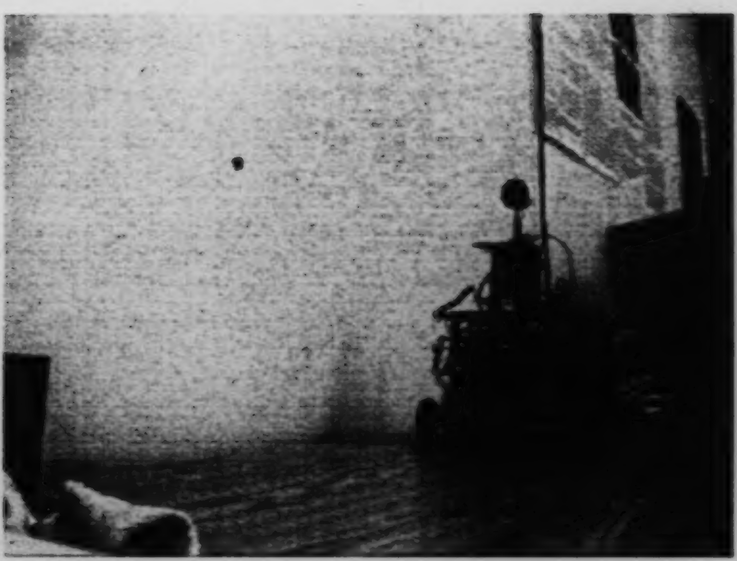
There are many image segmentation algorithms. Some of the techniques involve finding region boundaries (edges) and getting the regions from this information. Other techniques involve using the general groupings of pixel properties. A third set of techniques finds the region directly. This last set of algorithms include region growing from a single pixel and splitting and merging techniques, as in quadtree methods.

As in the edge detection algorithms, the most appropriate algorithms will provide segment images quickly, preferably in a single pass. Most of the algorithms involve multiple passes over the image data. In [72], the authors present a single pass algorithm to do image segmentation that is based on a partition mode test. This algorithm provides good images quickly, but remains too slow for this application with the available hardware.

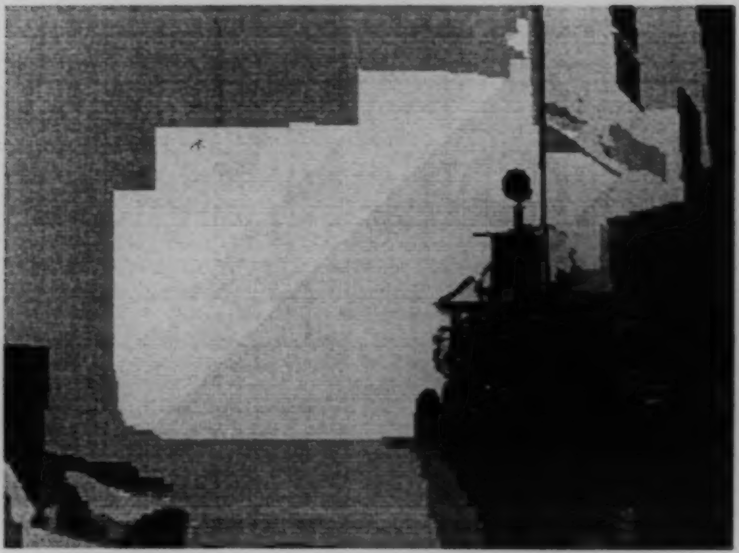
This segmentation algorithm uses a sliding 2 × 2 window in a single pass over the image. For each location of the 2×2 window, an examination of all the pixels in window is performed. Depending on the relative intensities of each of the pixels, a mode is assigned to the pixel group. Based on the mode, the pixels are assigned into segments and labels are updated if necessary (if pixels previously assigned to different segments are now determined to be in the same segment) and the previously unprocessed pixels are assigned an appropriate label.

Figure 7.6 shows an example image that has been passed through the segmentation algorithm. We can see that the segment image does lose detail, but details such as writing, wood grain, etc. are not important for the telerobotic tasks we are interested in. We retain details that are important for the tasks such as object outlines. Using the simple encoding scheme described in Section 7.2, we can compress the segment image from 307,200 bytes to 24,428 bytes, a compression ratio of 12.58. This size includes the grey-level for each set of pixels as well as the length of each run.

Segment image processing time is dependent on the input image to a small degree. The algorithm used to generate Figure 7.6 is a single pass algorithm, but at each pixel, there is potential for additional processing depending on whether the regions are to be merged. When the regions are being merged, the labels used need to be changed. This may not take too long, but it shows the dependency on the input image. An average 320 × 240 pixel image takes approximately 0.094 seconds to process. This is much slower than the edge detection algorithm and shows that the system is not ready to use segments as feature images.

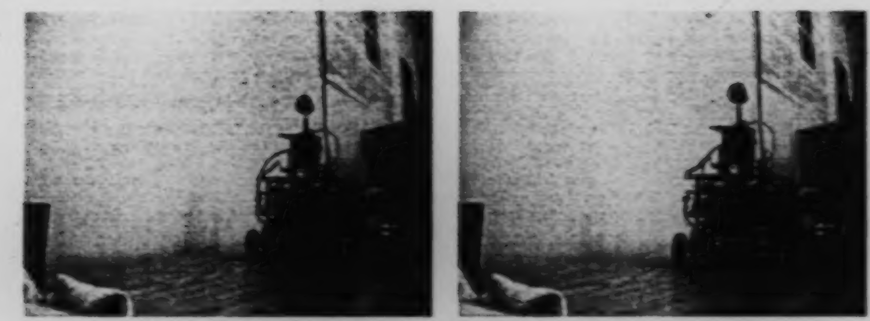


(a) Original image

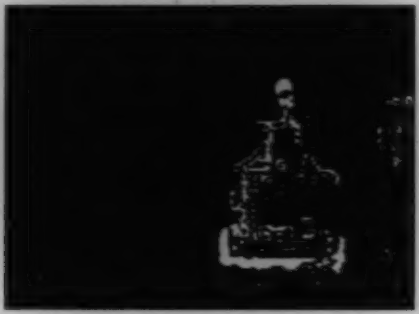


(b) Generated segment image

Figure 7.6: Example segment image. Original image size is 640 × 480.



(a) First image frame (b) Second image frame



(c) Generated difference image

Figure 7.7: Example motion image. Original size is 640 × 480.

7.4 Motion Images

Motion or difference images can be very helpful to draw the attention of an operator. When a static camera is observing a scene, a difference image will show anything that has changed since the previous frame. In a surveillance tasks, this can be helpful by drawing the operator's attention to a changing object or an approaching person. Unfortunately, in telenavigation, the telerobot and camera are usually moving. This means that the whole scene will change between frames and the resulting motion images will be saturated with motion. For static scenes, motion images can still be helpful.

Unlike the other types of feature images mentioned above, motion images cannot be obtained from a single pass on a single image. To obtain motion, at least two images and sometimes more are required depending on the technique.

There are essentially two sets of techniques for detecting motion in image sets. The first set of techniques are applied to images in the spatial domain. This means that the techniques are applied to the actual pixel values or some function of the pixel values. Techniques applied to the frequency domain are also possible. These techniques involve using the Fourier transform, which can be computationally expensive.

One of the simplest methods for determining motion images is to take the difference between two images. These two images do not have to be temporally successive, but when they are close in time, then the difference image will be less complex. A difference image can be mathematically defined as:

$$d_{ij}(x,y) = \begin{cases} 1 & \text{if } |f(x,y,t_i) - f(x,y,t_j)| > \theta \\ 0 & \text{otherwise} \end{cases}$$

where $d(x,y)$ is the difference image, $f(x,y,t)$ is the image at time t, θ is some threshold value and t_i and t_j are the times of the images being differenced.

This motion detection technique produces difference images. In Figure 7.7, there are two images taken from a stationary camera with a moving scene. These two images are used to produce the binary difference image in the same figure. Using the run-length encoding scheme on this image results in a byte stream of 6,808 bytes and a compression ratio of 45.12. Similar to edge images, this processing is not dependent on the input image. The average time of processing for this feature image type is 0.0029 seconds.

7.5 Chapter Summary

This chapter has presented different types of feature images and their application to very low bandwidth telenavigation tasks. Edge images provide the most exciting results and seem to be the most appropriate feature type for telenavigation. Segment images can also be effective in telenavigation tasks, but motion images are not appropriate when the camera is moving although other teleoperation tasks can benefit from their use.

Edge features can also be extracted as line segments so only the end points need to be transmitted to the display station. This can significantly reduce the amount of data that needs to be transmitted from every pixel on an edge to only the end points of the line segment. This is a promising avenue to pursue, unfortunately, current technology does not allow extraction of line segments using the Hough transform sufficiently quickly for the real-time constraints applied to the telenavigation tasks so the larger sized edge images are being used in the rest of the experiments.

The next chapter will present the concept of *Hybrid Video*. This allows the combination of the high rate feature images from this chapter with low frame rate video to provide additional information to the operator. Using hybrid video allows the operator to gain the fast updates of the features with the additional information, colour and texture for example, from full video.

Chapter 8

Hybrid Video for Varying Bandwidth Applications

In the previous chapter, feature images were presented as either an alternative or a preprocessing step for traditional compression techniques for very low bandwidth applications. The same feature images can be used as an enhancement for full video streams when there is more available bandwidth. In these situations, a combination of feature images transmitted at a high frame rate and full video at lower frame rate can be used to give better feedback to the operator of a teleoperation system.

Schilling and Cosman [67] present another approach to combining edge features and video to enhance the results for wavelet-based compression at very low bit rates. This approach transmits edge information with the image header and uses this information when reconstructing the original image from the wavelet coefficients to create "natural-appearing" images. This approach could provide a method of transmitting both the edge and full video information in our hybrid video approach if processing can be accomplished in the time allowed.

By using both high frame rate features and low frame rate video, we can provide several options to the operator. Displaying only the low rate video, allows the operator to have complete colour and texture information about the environment. For applications that have little or no time constraints, this information will allow the operator to complete their task properly, although possibly quite slowly. Displaying high rate features allows the operator to perform their task much faster, but requires the operator to have more knowledge about the environment and be able to interpret the features correctly.

Using a combination of both types of video information provides the operator with both possible advantages. They can view the high rate features to allow for fast updates of the environment they are operating in and the low rate video will maintain the colour and texture information required for high accuracy.

When the application has a time constraint, either because the task must be completed quickly or because the environment changes over time (which is the case of a moving robot), then using both types of video information provides a means to accomplish the tasks quickly and accurately.

Wire-frame model graphics combined with video data has been used in teleprogramming applications as well as augmented reality for teleoperation. Both of these applications show that using the added graphics improves the operator's performance under adverse conditions.

This chapter will discuss the theory and implementation of a prototype hybrid video teleoperation system. In the rest of this dissertation, the panoramic camera has been replaced with a limited field-of-view camera statically mounted on the robot. The resolution of the video system is better with the limited field-of-view camera. In Section 8.1, I will expand on the issues of bandwidth and combinations of full video and features. Section 8.2 will discuss different means of combining the video information. Finally, section 8.3 will discuss the different display methods that have been used in this research.

8.1 Hybrid Video

There are two extremes for available bandwidth that are of importance. The first is when there is very low bandwidth available. In this instance, displaying full video is very difficult. The previous chapter presents a method for displaying important feature information when the available bandwidth is very limited.

The opposite extreme is when there is significant bandwidth available.

Dedicated satellite connections are a good example of communication lines that provide very high bandwidth. On this extreme, full video can be transmitted at full frame rate without losing information. This demand on bandwidth can be reduced using compression techniques, but to maintain quality, the available bandwidth must still be high.

While the upper extreme can display a video stream at full frame rate, it too can benefit from using feature images to enhance the display. The lower extreme could use very slow full video frames, but the motion of the robot and the extremely low update of the full video stream would make the video texture unusable long before the next update.

In between these two extremes, using features and full video to complement each other will produce an environment that contains complementary information. There needs to be sufficient available bandwidth to accept the feature images (at least 25,000 bps) as well as some full video images. These images can be updated very slowly (every few seconds) or quickly depending on the available bandwidth.

In this system, the full video frames are not compressed. Compression will certainly allow more frames to be transmitted in the same time period. A single grey-scale, 8 bits per pixel image of 320 × 240 pixels requires 614,400 bits per frame. To add one full video frame per second to the feature image data, the system needs to have 639,000 bits per second transmission capacity.

While determining the number of full images transmitted based on bandwidth is theoretically possible, it is easier to alternate between full video frames and feature images based on a ratio. Throughout the rest of this section, this ratio will be referred to as the *frame ratio*. As the bandwidth becomes limited, a higher ratio is used to maintain the frame rate of the feature images as high as possible.

8.2 Combining Feature Images and Full Images

There are several interesting issues involved in combining feature images with full video images. The first is how different feature image types can be combined with the ordinary video stream. A common problem is registration of features with the video stream. If there is delay before displaying the video stream, the feature image that is being overlayed may come from a later video frame. There are several ways of working around this. The easiest is to ignore the registration problem, but that will cause problems for the operator if the difference between the feature image and the video stream is large.

Modeling the motion of the robot/camera system could be used to attempt to predict where the robot is in relation to where it was a certain amount of time before. Using this information, we can register the feature image onto the video stream but only display a possible small section of the feature image.

Another approach involves the predictive display of Chapter 6. In this approach, only a section of the full video is displayed at any time. The system predicts both the camera motion and the operator's viewing frame and requests the feature image that corresponds to the new scene based on these pieces of information.

Another issue for the combination of feature images with video streams is how to display the feature images. The first attempt uses an overlay technique. With edge images, we are trying to overlay a binary image on the video stream. This can be done quite simply by adding the two images together (thresholding to maintain the maximum pixel intensity). Figure 8.1 shows a single hybrid image created by overlaying the edge image from Figure 7.2 on its original, "raw" image. We can modify this technique slightly by incrementing the grey-level value of a pixel in the video stream that corresponds to an element of an edge in the edge image. There are many other methods of overlaying binary images, but most of them are quite simple. Motion images can be overlayed in the same manner as edge images as they can both be represented by a binary image.

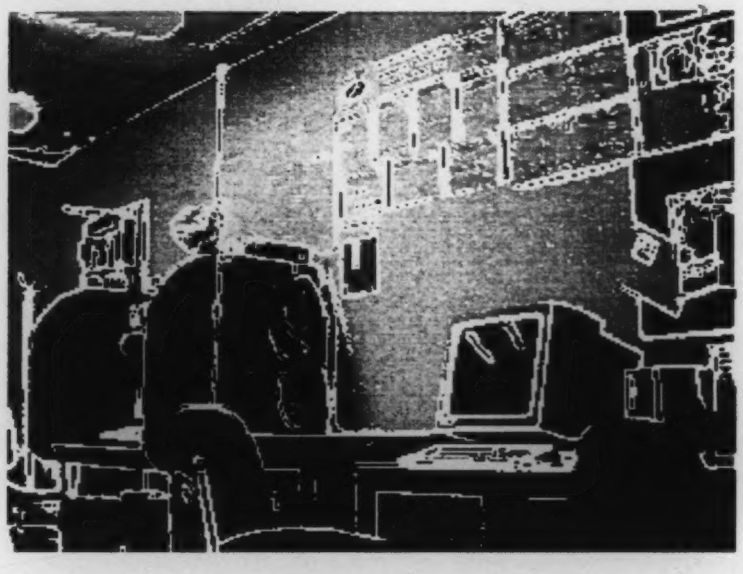


Figure 8.1: Example hybrid image overlaying edge features on video. Original size is 640 × 480. Generated from the images in Figure 7.2

Since segment images can contain large regions with a constant grey-level, using a simple ORing technique to overlay the segment image on the video stream will mask large portions of the image. One method that can be used to overcome this problem is to change the opacity of the overlayed feature image. A simple way of using opacity control is to combine the base image and the feature image using the following formula:

$$\mathbf{H} = k\mathbf{O} + (1 - k)\mathbf{B}$$

where **H** is the resulting hybrid image, **O** is the overlayed feature image, **B** is the base image, and k is the percentage of the overlayed image to be displayed, or the opacity of the overlayed image. This opacity controlling technique can also be used for the binary feature images, especially motion images because they can also contain large regions that can obscure the base image. Figure 8.2 shows an hybrid video frame using segment features.

Use of feature images as an overlay can help an operator with several issues that can be encountered with video streams. The first issue involves a delay in displaying the video stream. This delay can arise from a number of

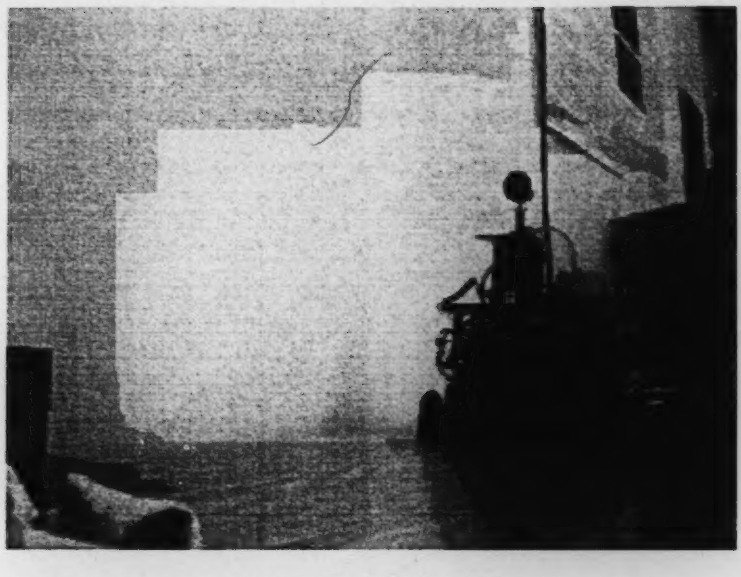


Figure 8.2: Example hybrid video frame overlaying segment features on the video frame. Original size is 640 × 480. Generated using opacity $k = 0.5$ from images in Figure 7.6.

sources including transmission or processing time. The use of feature images can help overcome the problems of delay by displaying feature images as a faster response to the operator's actions than the delayed video stream, or by displaying several frames of feature images while only one frame of the video stream is being displayed.

A second issue with video streams is the quality of the image frames. Image quality can be influenced by a number of factors such as visibility conditions at the remote site, compression of the video stream, or transmission problems (lost packets, poor connection, etc.). The main concern in this case is with enhancing the existing images to improve the operator's performance. These image quality issues can arise with or without the delay issues mentioned above.

8.2.1 Motion Estimation and Compensation

Motion of the robot can cause problems when registering the full video frame with the feature image. One way to aid in registering the two images is to estimate the motion of the robot. Since the only information available from the robot in this research is the images themselves, estimating motion must be done using this input.

The motion estimation is done using feature image frames. The frames used are the current frame and the frame received immediately after the most recent full video frame. Using these two frames, we can estimate the motion since the last full video frame was received and adjust its position accordingly.

While it is possible, in theory, to estimate the motion for all pixels in the pair of images if the depth of each pixel in the original image and the motion of the camera are both known, in practice, the exact knowledge is unavailable. Without this knowledge, a technique such as optical flow [35] can determine the motion of each pixel, but this can be computationally expensive.

There are several other techniques, less computationally intensive, that could be used for estimating motion. Much of the research is done in the context of feature matching for stereo vision applications. In stereo vision, matching is used in determining depth or distance. In these applications, the two viewpoints are relatively close to each other meaning that the matching feature should be close to the original position. These matching algorithms try to match features; for example, edges in [50] for robot manipulation or curvature in [15] to determine the 3-D pose of a flexible object.

The previous approaches attempt spatial matching, that is matching features that exist at the same time in different locations. For hybrid video, temporal matching must occur. This matching is between two images that are taken at different times. This matching generally uses block matching techniques. MPEG attempts to match small blocks between consecutive image frames when doing compression.

Other examples of block matching can be found in [26] and [59]. The authors of [59] use block matching for video conferencing compression. This algorithm determines the motion of small blocks which can then be used to compress the video stream, similar to MPEG and H.263. The authors of [26] describe a block matching algorithm to track objects moving in a video stream from a fixed camera. This algorithm assumes that the objects being tracked

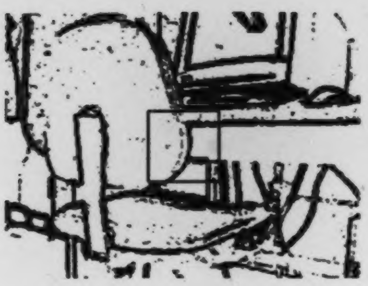


Figure 8.3: Location of block to be matched in original frame.

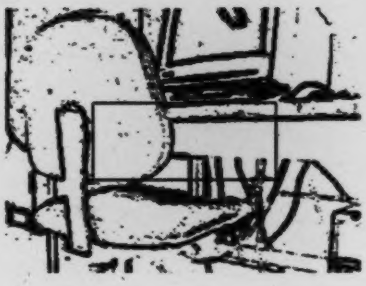


Figure 8.4: Search region for matching block

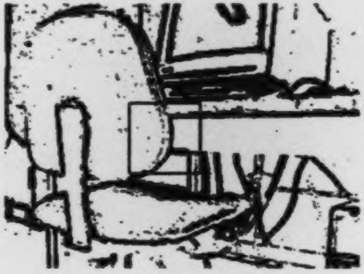


Figure 8.5: Matching block in second frame

remain within the camera view throughout the tracking.

In this system, the camera is moving, while those described above assume a static camera. The moving camera can cause serious problems. When the motion is simple translation, the images will not change quickly and the update rate for the full video images is sufficient to change the image before the motion has progressed so far as to make estimation impossible. The biggest problem occurs with rotation. The telerobot rotates quickly and within only a few frames, the scene can have changed sufficiently so the motion would place the full video image out of the viewing window.

The motion estimation approximation used in this research applies a block matching method. This approach assumes that the scene is static with a moving camera and that the motion can be approximated with a single two-dimensional motion vector. The system finds the best match of a single block in the earliest edge frame since the last full video frame (we will call this the original frame) with a position in the most recent edge frame. The block being matched is located in the centre of the original frame (Figure 8.3 shows where this frame is located). The second frame is searched in a long, thin region around the centre of the second image (see Figure 8.4). We search only this region because the motion of the robot constrains the motion in the image to be predominantly in the horizontal direction. Rotations of the robot translate into (potentially large) horizontal shifts of the images while forward or backward translation provide only small scaling changes in the subsequent images.

Since this processing must be integrated into the display or communication client, it needs to be done within the acceptable frame update period so that the processing does not slow down the display. The matching process has been tested and timed using both synthetic motion, generated by simply shifting an image and attempting to match to the original, and real images taken from the moving telerobot.

Table 8.1 summarizes some of the results from the matching process. The synthetic image was shifted thirteen pixels right and two pixels up. The translation images were taken from the telerobot moving forward in the lab environment. Finally, the rotation images were taken from a rotation to the left. These images were taken after the robot had rotated a very small amount, less than a standard period between transmission of edge images.

To reduce the time of the matching process, different amounts of subsampling occurred. This subsampling occurred in both images of a set. Since the edge images were generated using Sobel which produces thick edges, subsampling can occur without losing a large amount of edge information (see in Section 7.2.3 about the Hough transform and subsampling).

A second factor in the reduction was the location of the inspected blocks. Without this reduction, the matching process performs a brute force search of the entire expected strip to find a matching block. This search can be reduced by searching only a subset of the strip. This subset is determined by searching only those blocks that start at certain intervals. This is represented in the table by *Search 1* which is an interval of one and *Search 2*, an interval of two.

	Search 1		Search 2	
Synthetic	13,-2	0.14s	12,-2	0.069s
Translate	-17,0	0.13s	-18,2	0.068s
Rotate	-17,1	0.14s	-16,0	0.071s

(a) Subsampling factor 1

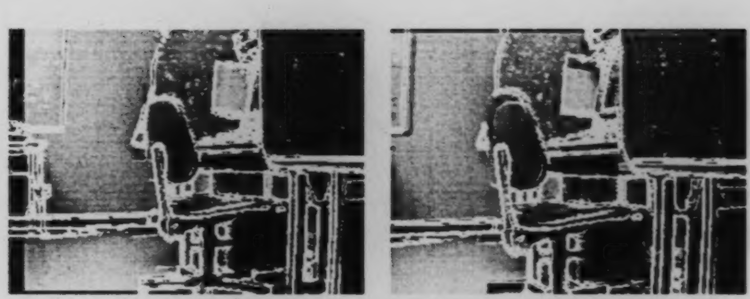
	Search 1		Search 2	
Synthetic	13,-2	0.069s	14,-2	0.037s
Translate	-17,2	0.069s	-18,2	0.036s
Rotate	-17,0	0.070	-16,0	0.037s

(b) Subsampling factor 2

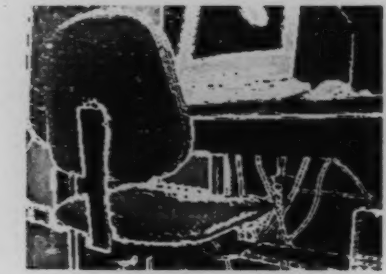
	Search 1		Search 2	
Synthetic	13,-2	0.046	14,-1	0.025s
Translate	-18,0	0.049s	-18,0	0.024s
Rotate	-17,-1	0.047	-18,0	0.025s

(c) Subsampling factor 4

Table 8.1: Summary of motion estimation and timing using three different image sets. Subsampling rate and search interval size correlate to a motion and timing pair.



(a) Synthetic motion.
(b) Translational motion.



(c) Rotational motion.

Figure 8.6: Overlayed images from matching process. Original images are 320 × 240 pixels.

The internal elements of the table show the estimated motion in pixels, both horizontal and vertical components, and the time to complete the matching. This is shown for all combinations of subsampling rate and search interval for the three image types. For example, using the synthetic image, a search interval of two and a subsampling rate of 1 show that the estimated motion is 13 pixels to the right, two pixels up and the time to find this match was 0.069 seconds.

The table shows that the matching process remains quite accurate even as the subsampling and the searching interval increase. The estimated motion remains within one or two pixels in either direction regardless of the parameter changes. Some of the difference can be accounted for by quantization error due to changes in the search interval. The synthetic image was intentionally shifted to show that the matching occurs as close as possible.

Figure 8.6 shows the results of the matching. Each image is composed of the full image shifted by the results of the fastest matching along with the second edge image related to that match, the edge image which has moved. From this figure, it is obvious that the matching process is sufficiently accurate to keep the full image moving with the edge images as they change over time as long as the motion remains small enough to keep the images in the search range.

Matching the translated images shows the worst results. This is due to a greater change in vertical position and the change in scale associated with forward-backward translation. As the camera approaches an object, it gets larger in the image. The scaling is not taken into account by the matching process, but such scale changes are small between full video frame updates.

8.3 Displaying Hybrid Video

The original display technique for our hybrid video stream was to simply overlay the full video image on the feature image. This was done with no attempt to register the two scenes. While this provides texture details to the operator, as the scene changes in the feature image, it does not change in the full video

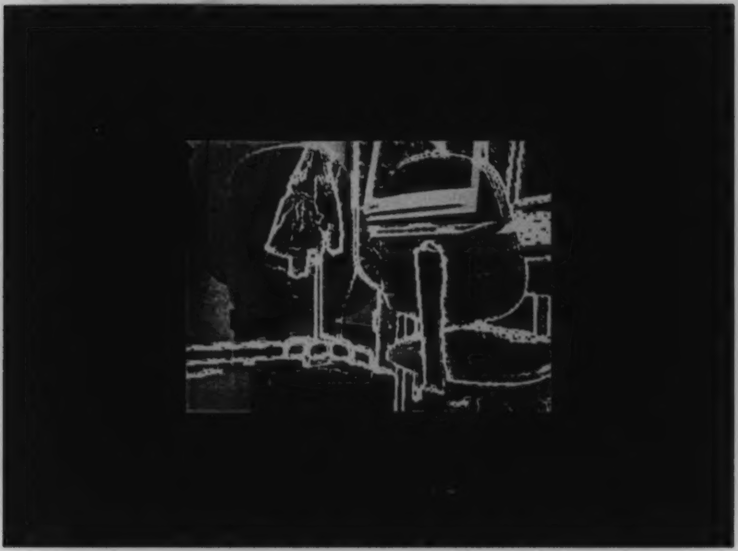


Figure 8.7: Hybrid image where the robot has rotated since the full video image was recovered

and the viewed scene becomes very difficult to interpret. Figure 8.1 shows a hybrid image where both the edge image frame and the full video frame are properly registered (the edge image was derived directly from the video frame). Figure 8.7 shows another hybrid video frame. This figure shows a scene where the edge image has been changed since the full image was taken (the robot was rotated) without any motion compensation.

The second method of displaying hybrid video involves using the motion estimation and compensation described in the previous section. In this display, the full video image is offset based on the estimated motion of the robot. Figure 8.8 shows an example of how this appears to the operator (the display uses the matched image from Figure 8.6(c)).

When motion is significant or when the motion estimation process cannot find a good match for the block, no motion is estimated and hence no compensation is performed. This lack of compensation generates images that are similar to those without motion estimation (as described above). In this case, operators are also shown confusing images.

One solution for the problem of badly registered images involves changing

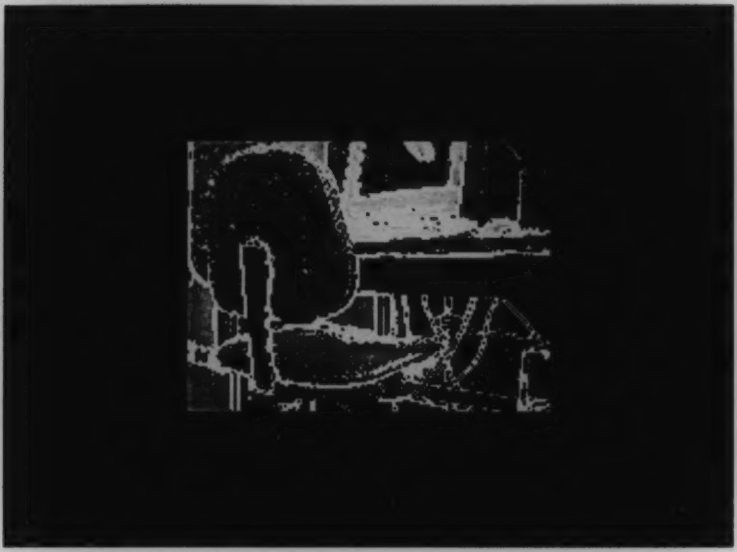


Figure 8.8: Hybrid video interface with motion compensation

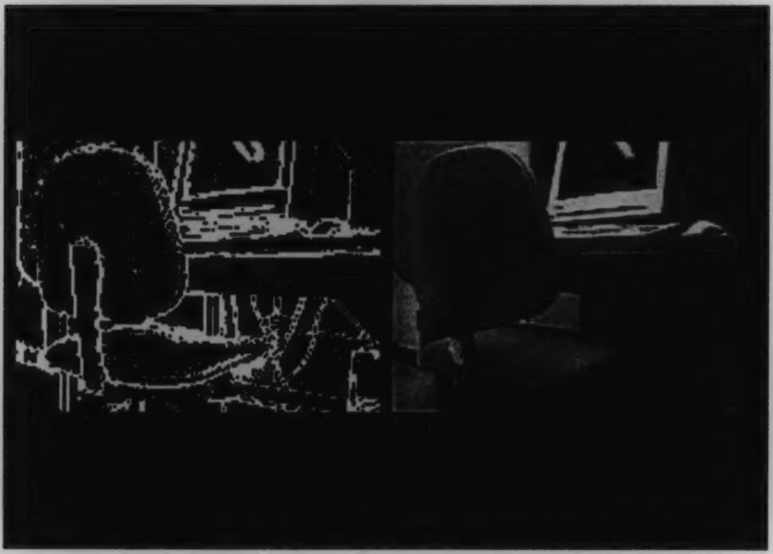


Figure 8.9: Hybrid motion using double image window

the display. In the previous solutions, the display overlays the edge image on the full video image. This generates confusing images when the motion is high or no estimation is performed. If the two images, the feature image and the full video image, are placed side by side, then the confusion derived from two conflicting images in the same space is removed. The human operator can watch only one of the two images at a time although he can switch between them quickly. Figure 8.9 shows an example of this display technique.

One drawback to this display technique is that it is slower to display image frames. In the single window technique, the two images can be easily combined in memory and then displayed. This requires only a single small change in the display window. In the double window process, each image must be displayed separately. This means that there are two sets of changes made in the display window. If no motion estimation is being done, then the display changes based on the full video image only need to be done when the full video has been updated saving some time, but then the benefits to using motion estimation are lost. Using the GL interface, the double window updates the images at 13.04 frames per second when using hybrid video.

8.4 Summary

This chapter has presented the ideas behind hybrid video. By combining a high rate feature image stream with a lower frame rate full video stream, the operator can be provided with a more informative display than feature images alone. The overhead of adding feature images does not reduce the frame rate of the video stream significantly, and the information gained more than offsets the slight reduction of the full video stream. Different methods of displaying the hybrid video were proposed, including overlay with or without registration and side-by-side display. The side-by-side display is used in future experiments to allow the operator to chose the video stream they will view without the conflicts that can occur if the registration process makes a mistake.

Chapter 9 will discuss the results of human experiments comparing hybrid video, edge images, and full images. The operators perform two different tasks for each of the video streams. These tasks are a navigation task and an object identification task. Each is performed using all of the different video streams to compare the ease of use of each.

Chapter 9

Performance of Full, Feature, and Hybrid Video

The previous chapters have described methods to display different types of video information based on available bandwidth and the tasks to be performed. Initially, feature images were introduced as a means of reducing the bandwidth requirements for very low bandwidth teleoperation applications. Following these feature images, this dissertation demonstrated that a combination of feature images and full video can also provide an environment which allows for teleoperation under different bandwidth limitations.

Since this research is based on telenavigation, experiments involving human operators are necessary to show that the proposed techniques provide sufficient information for successful teleoperation. For these experiments, only edge feature images are tested. These features have been chosen because of their importance in navigating in an indoor environment. Based on the implementations discussed previously, generating and transmitting edge images also provides the fastest frame rate of all the feature images discussed.

The experiments discussed in this chapter are executed several times. The two tasks, navigation and object identification, are performed by the operator for each of three displays.

The first display involves a full video image stream. This stream is captured directly from the camera while the only processing is to crop the images from 640 × 480 pixels down to 320 × 240. Prediction could have been used, but was removed from the interface during this set of experiments to reduce the complexity presented to the operator.

The second display type uses edge images. These images are captured from the camera and generated using the Sobel operators discussed previously. During the experiments, the operators are allowed to change the threshold used in the edge detection algorithm, in both this and the hybrid video display. In practice, once the experiment has started, the operators do not change this value from the initial setting.

The final display is a hybrid video display. The experiments use the side-by-side display techniques described previously. In order to keep some parameters constant for all operators, the frame ratio is preset to 10 edge frames for every full video frame.

This chapter will present the results of experiments designed to show that these techniques allow an operator to perform teleoperation tasks with the same rate of success as if they used purely video information. Section 9.1 will discuss the design of the experiments. In Section 9.2, the results of the experiments will be described along with some of the comments that were collected.

9.1 Structure of the Experiments

All three types of video streams need to be tested in such a way that they can be compared. We know that it is possible to perform a telenavigation task, but we do not know how easy it is to perform the task using different display techniques.

The tasks the operators have to perform are two fold. They involve a navigation aspect and an object identification aspect. In an indoor environment, these two tasks are important in the operation of a telerobot.

For mobile telerobots, the ability to navigate in the environment is fundamental. If no navigation is required, why use a mobile robot? Surveillance applications, moving objects from one location to another, or simply getting a manipulator to a certain location, all require navigation capabilities. The operator must be able to see where they are going and know when they get where they need to be.

Object identification is also an important aspect in mobile telerobotics. How does the operator know that they are in the correct location for their manipulators to perform the appropriate action? How does the surveillance expert know that an office has been broken into? In both cases, the operator identifies an object and can then take the appropriate action.

The experiments are designed to use two workstation in the laboratory as the base station and operator display. The base station is a Pentium workstation running RedHat Linux kernel version 2.2.14 and the operator display station is an SGI Indy using IRIX version 6.2. The two workstations are connected using an Ethernet network. This network provides approximately 850 KBps transmission rate. This network supplies much greater bandwidth than an analog telephone line, but since no compression is being used, we can still provide good transmission rates for the different video streams without significant degradation in image quality.

For these experiments, the viewing camera has been moved from the top plate on the robot to the bottom plate. This allows the operator to have a better view of low objects and avoid those that would interfere with the wheels. Since the objects in the environment are chairs, tables, a tripod, garbage cans, etc., viewing them close to the floor allows the operator to see their legs and wheels instead of the upper portions. Usually it is these lower portions that extend out and a high camera position will miss them.

The experimental environment is designed so the operator is unable to see the robot directly. The only information provided to the operator is that which can be seen through the camera mounted on the lower plate.

9.1.1 Navigation Experiment

Both experiments contain a navigation component. This section describes the experiment that is concerned solely with navigation. In any mobile application there will be a navigation component.

In the pure navigation experiment, the operator is asked to navigate the telerobot in a cluttered indoor environment. The operator is asked to navigate the telerobot through the environment avoiding obstacles to the best of their ability. The success of avoiding obstacles will be discussed with the problems in Section 9.3.

In this task, the telerobot is started in the lab at a known location near the operator's display terminal. The operator moves the telerobot from its starting location to the end of the lab near the doorway. This involves navigating in a generally straight line while avoiding obstacles, such as chairs and tables, on the sides of the path. The operator then should turn the robot approximately 90 degrees to the right and drive through the doorway into the corridor. Once into the corridor, another 90 right turn is performed and the robot proceeds down the corridor.

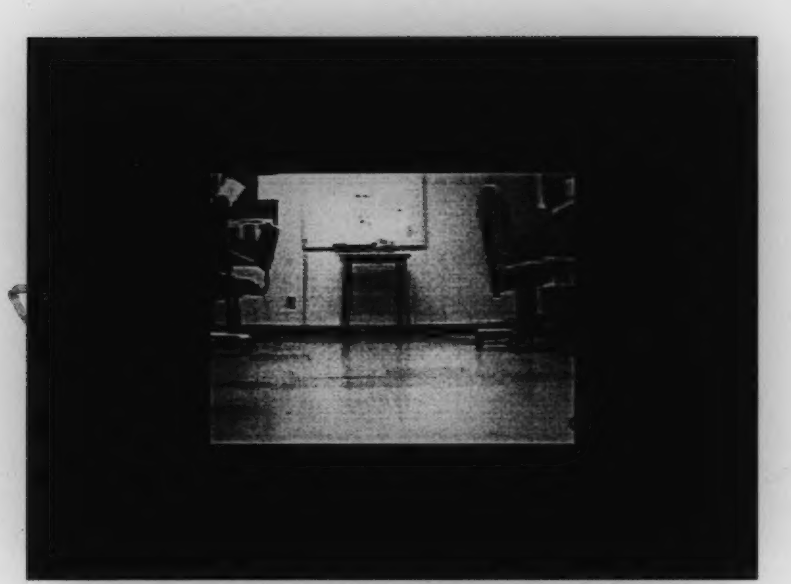
After the telerobot proceeds down the corridor a short way, to ensure the operator has recognized they are in the corridor, the robot is turned around and returns to the lab through the doorway reversing its previous course to return to the end of the lab (not the starting point). The pure navigation task ends when the robot has re-entered the lab. At this point the object detection task is begun. This will be described in the next section.

Figure 9.1 shows the view from the camera when the robot is in its starting position. This figure shows both the full video image of the scene and the edge image representation. The hybrid video display will show both images side-by-side.

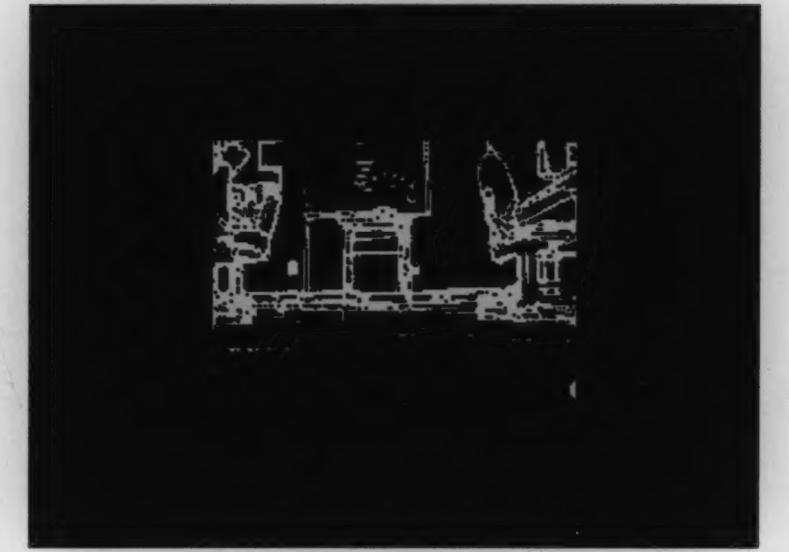
9.1.2 Object Identification Task

In the object identification task, the operator is asked to locate and approach an object they are shown before the experiments begin. Figure 9.2 shows the object the operator is searching for. The operator is shown the actual object at the beginning of the experiment rather than an image of the object. This may have added to the difficulty of identifying the object using the edge images, but the two images in Figure 9.2 show that identification using edges is not significantly more difficult.

When the navigation task described in the previous section is complete, i.e. the robot has returned to the lab, the operator is told to begin the object

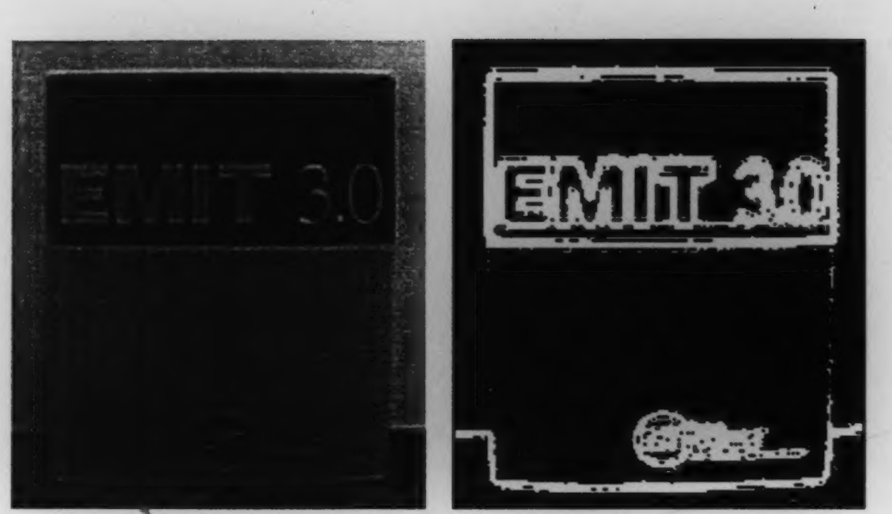


(a) Full video.



(b) Edge video.

Figure 9.1: Starting view from the robot camera.



(a) Full box.

(b) Edge image of box.

Figure 9.2: Object in identification task.

identification task. The object has been placed somewhere in the lab environment between the initial starting point of the robot and the point where the robot ends the navigation task. The object is located somewhere on the sides of the path the robot originally took to pass through the lab in the first task.

The object has been placed on or near the floor, within five centimeters. When the telerobot is looking at the object, it is within the field of view. It may be hidden from certain angles, but the telerobot can always view the object from somewhere in the lab environment.

The operator must navigate in the environment, a rectangular area in the middle of the lab, and find the object. Once they believe they have found the object, they are to move the telerobot to within one metre to indicate that they have found the correct object. When the object has been correctly identified, the telerobot is returned to its initial position to prepare for the next operator or display technique.

At this point, the operator is asked to answer several questions relating to the difficulty of performing the tasks using the display. These questions are discussed in the following section.

9.2 Evaluation of Performance

The evaluation criterion that is of interest to this work is whether the operator can perform the tasks required using the various displays. To evaluate this, each operator is asked the same set of questions after each iteration of the navigation and object identification tasks.

Before attempting the experiments, each operator is allowed some time to become familiarized with both the controls for the telerobot and the viewing environment. None of the operators have previously used the telerobot. Several had participated in the prediction experiments (discussed in Chapter 6), but that did not involve moving the telerobot.

When the tasks have been completed, the operator is asked to rate the difficulty (very easy to very difficult and impossible) of performing the task using the display. They are also asked to explain their rating and for the hybrid video display, how much time was spent looking at each subwindow. This information allows us to determine a comparison for that operator for all of the displays.

Since each operator is different, comparing between operators does not provide sufficient meaningful information. One operator may find all tasks easy, but require much longer time than a second operator who may find the full video requires the longest time and is the most difficult to use.

The operators performing these experiments were selected from the graduate students and research associates working in the laboratory. These participants are all used to working with computers and robots, however they are unused to performing teleoperation tasks. Five operators are used in these experiments although the fifth was unable to perform the experiments using hybrid video due to hardware problems (these are discussed in Section 9.3).

Figures 9.3 and 9.4 show the operator ratings for the navigation and object identification tasks. The operators are shown across the horizontal axis, numbered one through five, while the difficulty rating is along the vertical axis. The difficulty is rated from one to six where one is considered impossible, two is very difficult, and six is very easy.

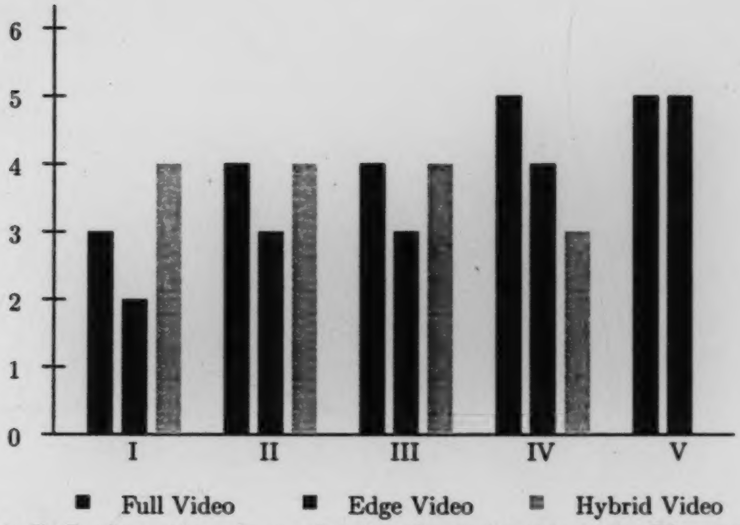


Figure 9.3: Difficulty ratings for performing the navigation task. The operator number is found across the horizontal axis with the difficulty rating along the vertical. The ratings for each operator are grouped above their number.

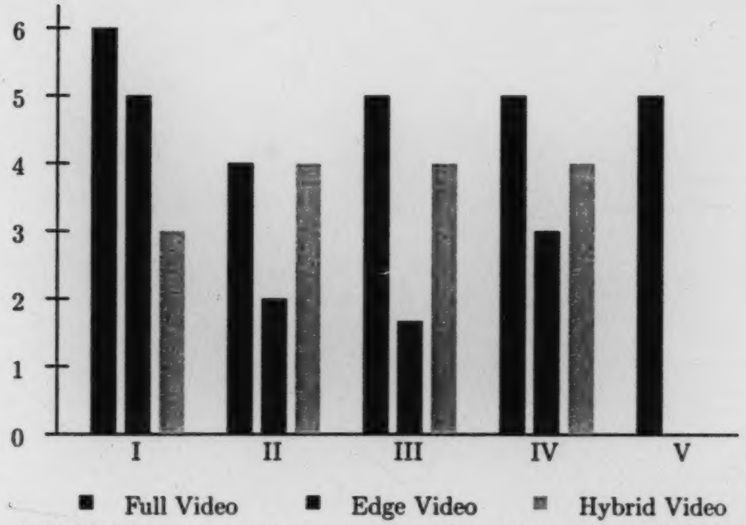


Figure 9.4: Difficulty ratings for performing the object identification task. The operator number is found across the horizontal axis with the difficulty rating along the vertical. The ratings for each operator are grouped above their number.

Each operator has rated the difficulty of all six parts of the experiment, each task performed once for each video stream. The exception is for operator number five who only completed a few of the parts and finished prematurely due to mechanical difficulties. The graphs show the ratings for each part grouped by task and then operator. For example, Figure 9.3 shows the ratings for the navigation task with the ratings for each operator located over the appropriate number.

In the navigation subtask, all operators found using the edge images to be either more difficult than, for four of the five, or as difficult as, one, the full video stream. In no case was the edge image display considered more than one step more difficult. For the operator, number five, who reported that both were of equal difficulty, the reasons given for the rating of the edge image experiment was "quite informative and first experiment is helpful". This indicates that this operator found the initial full video experiment to be more difficult probably due to insufficient time taken to become familiar with the system.

The hybrid video portion of the navigation task was perceived to be easier than simply using the edge images, with one exception. One of the four operators who completed this portion of the experiment found that the hybrid video navigation experiment was more difficult than the edge image experiment. This is probably because the operator "most of the time" used the low frame rate full video image. Why she chose to pay more attention to this image is unknown.

For the object identification task, all operators found using edge images to be more difficult than full video. This change in difficulty ranged from one to three steps. The operators agreed that the lack of intensity information and trying to locate an object by shape, when other objects have a similar shape, made this task difficult.

The hybrid video experiment portion of this task was generally considered easier than using edge images. The one exception was the first operator who found this more difficult due to the location of the object. In this particular experiment, the object was placed so that it was hidden from the direction the robot was facing at the beginning of this portion of the experiment.

In comparing the hybrid video object identification task to the full video task, three of the four operators found hybrid video slightly more difficult. The fourth operator found both experiments to be of equal difficulty.

Only three of the operators gave an idea of how much they used the edge images and the full video during the hybrid video experiments. Two of these operators used the low rate full video most of the time while one used the two image streams equally. It appears that even if the full video is being displayed at low frame rate, the images were updated about once per second, some operators will always use this information even though the edge image is being updated ten times faster.

9.3 Problems Encountered

During the course of the experiments described in the previous section, several problems arose. Some of these problems were such that experimentation needed to be stopped (the hardware problems). Before any further experiments can be conducted, these hardware problems need to be addressed.

9.3.1 Collisions

Many operators ran into objects. These collisions were due to several factors. The first factor in collisions was a lack of depth perception. Without depth perception, operators were unable to judge the distance to an object and would suffer a head-on collision. The most common collision was with a door during the edge image experiments. The door did not have many distinguishing edge features causing it to vanish from the display when the robot was too close.

A second reason for colliding with objects was the narrow field of view provided by the camera. Operators unused to the telerobot are not accustomed to the width of the robot and are unsure of its location with respect to objects that have left the camera view. This leads to collisions with objects on the side of the robot. The most common collision was with the door frame.

A final minor type of collision was due to the height of the robot. Because the camera was placed on the bottom of the robot, the operator remained unaware of objects that approach the top. Fortunately, there was only one object in the environment that can cause this problem and it could be generally avoided by avoiding its legs. This was a problem with the experimental environment.

9.3.2 Robot Control

For these experiments, the robot was controlled by the keypad. When a key was pressed, the robot would perform that action, but when the key was released the robot would stop. This provides a easy means of stopping the robot, but the event queue used for key presses and releases caused a problem. This problem was that the robot tended to stop and start. The event queue picked up key releases when there were none. These phantom key releases were not filtered out and caused the robot to stop more frequently. This software problem caused some of the hardware problems to recur frequently.

9.3.3 Hardware Problems

There were several hardware problems encountered. Most of these were identified or made worse by the robot control problem discussed above. Each of these problems was either fixed or could be temporarily repaired. Unfortunately, as the experiments continued, the problems recurred more frequently and eventually became too frequent to continue. Fortunately, by this point, sufficient data had been collected. The hardware problems included insufficient strain relief on the camera cabling and loosening of the wheels and the axle couplings.

9.4 Summary

This chapter has presented the experiments using human operators that compare the different video streams. These experiments involved both navigation and object location tasks. Each operator performed both tasks while using three different video streams: edge images, hybrid video with edges, and full video. It has been shown that both the edge image and hybrid video streams provide sufficient information to allow the operator to complete the tasks successfully although the object location task becomes more difficult when edges are the only information provided.

Chapter 10

Conclusions and Future Work

This dissertation has presented work designed to reduce the amount of bandwidth required for teleoperation applications. This reduction is necessary when the application uses a network where the available bandwidth is severely limited.

The first proposed technique involves predicting operator input. In this dissertation, the teleoperation system involves using a mouse to control a viewing window. The viewing window reduces the size of the image that the operator uses to control the telerobot but also allows the operator to change the point of view and see more of the environment.

When the camera system uses a panoramic viewing system in place of a traditional pan-tilt unit, the operator has access to the full environment without delays related to the mechanical system. By using a viewing window which only contains a fraction of the full panoramic image, the operator is presented with sufficient information to operate the telerobot without the overwhelming, complete panoramic view. Predicting where the operator is viewing during the next time period provides an apparent reduction in the delay in the image stream transmission.

In the predictive window approach, a Kalman filter is used to predict mouse motion to reduce apparent delay in visual display. This can provide better results than transmitting and displaying only what the camera is currently viewing. Using models of mouse motion that include the simple constant velocity model, the constant acceleration model, and the complex friction model give more accurate results, closer to the real motion values, than the "no prediction" model, which assumes that the mouse does not move before the next time step. This has been shown to be true for controlled mouse trajectories and for real operator task-based trajectories in general.

It has also been shown that for a limited field-of-view location task, operators either prefer a prediction method, or have no strong preference of predictor. For tasks involving the panoramic video system, this predictive window approach allows the operator to have generally better performance in an object tracking task: the perceived quality of the video stream is better with the predictive display, as is the quality measured by not having to display old or blank information.

This technique allows the teleoperation system to transmit only a portion of a large image that an operator is viewing, along with the portion that he is predicted to be viewing in the near future. This predictive window technique allows a first reduction in the bandwidth required for some telepresence tasks, as well as improves the quality of video feedback when only a section of the image can be viewed.

The second technique to reduce the amount of bandwidth required is using feature images. These feature images can be used in applications where there is only very low bandwidth available such as over the Internet or a traditional phone line. Feature images can be used as a preprocessing step before using traditional compression techniques or using compression specific to the feature images, such as run-length encoding for binary edge images.

High frame-rate feature images can be combined with a low rate video stream when the available bandwidth is high enough that the feature images are being transmitted at full capacity with more bandwidth available. This allows additional information, such as colour or texture, to be transmitted along with the important features giving the operator more details of the remote environment.

Both edge images and hybrid video using edges provide a telerobot operator with sufficient information to perform indoor navigation tasks. The operator can also perform object identification tasks although these tasks are more difficult. Operators with more experience should be able to learn to use the high rate feature images for the navigation portion of combined tasks.

This dissertation has shown that using these approaches, even without optimized image processing and feature image sizes, an operator can still successfully perform these telenavigation tasks. With improvements to this approach, teleoperation can be performed with nearly any amount of available bandwidth as long as it can be carried by a phone line. This will allow teleoperation from home without the need of special technology.

10.1 Future Work

While this dissertation shows that these approaches can reduce the amount of bandwidth necessary for successful teleoperation, there are several areas that can be improved. The first is the performance of the interface itself. The interface as it stands uses an inefficient approach to processing and transmitting the images. Improving the image delivery should drastically improve the performance of the system.

In the prediction process, the input has assumed a simple physical model of a mouse (involving friction). There are other models that could be used in the prediction process – for instance, utilizing different models for the mouse, or extending the research to use different input devices, such as a head-tracking system. The research presented here uses low-resolution imaging systems: the standard limited field-of-view NTSC camera, and the panoramic imaging system, which also uses an NTSC camera. There are much higher resolution systems available which could dramatically improve the quality of the images presented to the user, with a higher transmission cost. With a higher resolution imaging system and a larger viewing window, it would be possible to transmit a smaller fraction of the image and still improve the overall quality of the information presented to the user.

For the feature images, further work can look at other feature types and include segment or motion images in hybrid video. The features will be dependent on the application, but different applications can be researched. Generating edge lists of appropriate edges sufficiently fast is another area that should not be ignored.

This dissertation has presented several methods for creating the hybrid video display: overlay without registration, overlay using motion estimation, and side by side display. Are there other methods that can be used, such as alternating frame types at high speed or registering small sections of the full video? The motion estimation presented uses a simple single block matching algorithm to estimate the motion of the robot. There are other approaches that could be investigated. Mobile robots can also mount sensors that can measure their motion and this data could be combined with the video based motion estimation, perhaps using a Kalman filter, to generate more accurate estimates.

The experiments involving human operators need to be expanded significantly. Only a small selection of graduate students has been used as the experimental population. This group should be expanded to include a large range of computer experience. The development of a set of subjective criteria is also a necessary part of any teleoperation research. Different teleoperation tasks will require a slightly different set of criteria, but a general overview will assist us in designing experiments using different tasks.

Bibliography

[1] F. Arai, M. Tanimoto, T. Fukuda, K. Shimojima, H. Matsuura, and M. Negoro. Multimedia tele-surgery using high speed optical fiber network and its application to intravascular neurosurgery. In *Proc. of 1996 IEEE Int. Conf. on Robotics and Automation*, volume 1, pages 878–883, Minneapolis, Minnesota, 1996.

[2] R. T. Azuma. A survey of augmented reality. *Presence*, 6(4):355–385, August 1997.

[3] J. Baldwin, A. Basu, and H. Zhang. Modeling and compensating for delays in telepresence applications. In *World Automation Congress*, Anchorage, Alaska, USA, May 1998.

[4] J. Baldwin, A. Basu, and H. Zhang. Predictive windows for delay compensation in telepresence applications. In *Proc. of 1998 IEEE Int. Conf. on Robotics and Automation*, pages 2884–2889, Leuven, Belgium, May 1998.

[5] J. Baldwin, A. Basu, and H. Zhang. Panoramic video with predictive windows for telepresence applications. In *Proc. of 1999 IEEE Int. Conf. on Robotics and Automation*, pages 1922–1927, Detroit, Michigan, USA, May 1999.

[6] A. Basu and K. J. Wiebe. Enhancing videoconferencing using spatially varying sensing. *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans*, 28(2):137–147, 1998.

[7] A. K. Bejczy, P. Fiorini, W. S. Kim, and P. Schenker. Toward integrated operator interface for advanced teleoperation under time-delay. In *IROS '94*, volume 1, pages 563–570, 1994.

[8] A. K. Bejczy, W. S. Kim, and S. C. Venema. The phantom robot: Predictive displays for teleoperation with time delay. In *Proc. of 1990 IEEE Int. Conf. on Robotics and Automation*, pages 546–551, Cincinnati, Ohio, May 1990.

[9] D. Bertsekas and R. Gallager. *Data Networks*. Prentice Hall, second edition, 1992.

[10] T. Boult, R. Micheals, A. Erkan, P. Lewis, C. Powers, C. Qian, and W. Yin. Frame-rate multi-body tracking for surveillance. In *Proc. of DARPA IUW*, November 1998.

[11] T. Boult, C. Qian, W. Yin, A. Erkin, P. Lewis, and R. Micheals. Applications of omnidirectional imaging: multi-body tracking and remote reality. In *Proc. of IEEE WACV*, October 1998.

[12] S. M. Bozic. *Digital and Kalman Filtering.* Edward Arnold, second edition, 1994.

[13] M. Bozorg, E. M. Nebot, and H. F. Durrant-Whyte. A decentralized navigation architecture. In *Proc. of 1998 IEEE Int. Conf. on Robotics and Automation*, pages 3413–3418, Leuven, Belgium, May 1998.

[14] K. J. Bradshaw, I. D. Reid, and D. W. Murray. The active recovery of 3d motion trajectories and their use in prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(3):219–234, March 1997.

[15] J.-E. Byun and T. Nagata. Determination of 3-D pose of a flexible object by stereo matching of curvature representations. In *IROS 94*, pages 1992–1999, 1994.

[16] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[17] C. Cauchois, E. Brassart, C. Drocourt, and P. Vasseur. Calibration of the omnidirectional vision sensor: Syclop. In *Proc. of 1999 IEEE Int. Conf. on Robotics and Automation*, pages 1287–1292, Detroit, Michigan, May 1999.

[18] S. Clark and H. Durrant-Whyte. Autonomous land vehicle navigation using millimeter wave radar. In *Proc. of 1998 IEEE Int. Conf. on Robotics and Automation*, pages 3697–3702, Leuven, Belgium, May 1998.

[19] C. Cooke and S. Stansfield. Interactive graphical model building using telepresence and virtual reality. In *Proc. of 1994 IEEE Int. Conf. on Robotics and Automation*, volume 2, pages 1436–1440, 1994.

[20] L. Delahoche, C. Pégard, E. M. Mouaddib, and P. Vasseur. Incremental map building for mobile robot navigation in an indoor environment. In *Proc. of 1998 IEEE Int. Conf. on Robotics and Automation*, pages 2560–2565, Leuven, Belgium, May 1998.

[21] C. Drocourt, L. Delahoche, C. Pégard, and A. Clerentin Gracsy. Mobile robot localization based on an omnidirectional stereoscopic vision perception system. In *Proc. of 1999 IEEE Int. Conf. on Robotics and Automation*, pages 1329–1334, Detroit, Michigan, May 1999.

[22] S. Emura and S. Tachi. Compensation of time lag between actual and virtual spaces by multi-sensor integration. In *Proc of 1994 IEEE Int. Conf. on Multisensor Fusion and Integration for Intelligent Systems*, pages 463–469, October 1994.

[23] S. Emura and S. Tachi. Sensor fusion based measurement of human head motion. In *IEEE International Workshop on Robot and Human Communication*, pages 124–129, 1994.

[24] S. Emura and S. Tachi. On design of sequential sensor fusion system. In *Proc. of 1998 IEEE Int. Conf. on Robotics and Automation*, pages 3400–3406, Leuven, Belgium, May 1998.

[25] E. Fabrizi, G. Oriolo, S. Panzieri, and G. Ulivi. Enhanced uncertainty modeling for robot localization. In *World Automation Congress*, 1998.

[26] J. Ferruz and A. Ollero. Visual tracking of mobile objects. applications in mobile robotics. In *World Automation Congress*, 1998.

[27] J. Funda, T. S. Lindsay, and R. P. Paul. Teleprogramming: toward delay-invariant remote manipulation. *Presence*, 1(1):29–44, 1992.

[28] J. Gluckman, S. Nayar, and J. Thoresz. Real-time omnidirectional and panoramic stereo. In *Proc. of Image Understanding Workshop, IUW98*, 1998.

[29] K. Goldberg, M. Mascha, S. Gentner, N. Rothenberg, C. Sutter, and J. Wiegley. Desktop teleoperation via the world wide web. In *Proc. of 1995 IEEE Int. Conf. on Robotics and Automation*, volume 1, pages 654–659, 1995.

[30] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Addison-Wesley Publishing Company, 1993.

[31] B. S. Graves. *A Generalized teleautonomous architecture using situation-based action selection*. PhD thesis, Texas A&M University, 1995.

[32] Moving Picture Experts Group. The mpeg home page. http://drogo.cselt.it/mpeg/.

[33] S. R. Gunn. On the discrete representation of the laplacian of gaussian. *Pattern Recognition*, 32:1463–1472, 1999.

[34] K. Hewitt. Desktop videoconferencing products. http://www3.ncsu.edu/dox/video/products.html.

[35] B. K. P. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[36] D. H. Hubel. *Eye, Brain, and Vision*. Scientific American Library, 1988.

[37] I. W. Hunter, T. D. Doukoglou, S. R. Lafontaine, P. G. Charette, L. A. Jones, M. A. Sagar, G. D. Mallinson, and P. J. Hunter. A teleoperated microsurgical robot and associated virtual environment for eye surgery. *Presence*, 2(4):265–280, 1993.

[38] TelePhotogenics Inc. Telephotogenics home page. http://www.telephotogenics.com/main.htm.

[39] C. Jaynes and J. Hou. Temporal registration using a kalman filter for augmented reality applications. In *VI 2000*, pages 1–8, 2000.

[40] P. Jensfelt and H. I. Christensen. Laser based pose tracking. In *Proc. of 1999 IEEE Int. Conf. on Robotics and Automation*, pages 2994–3000, Detroit, Michigan, May 1999.

[41] K. Kim, M. W. Vannette, J. L. Hall, and Flugrad D. R. Remote manipulator and its teleoperated guide vehicle for hazardous waste sampling operations. In *Proc. of 1994 IEEE Int. Conf. on Robotics and Automation*, pages 2620–2625, 1994.

[42] A. Kiruluta, M. Eizenman, and S. Pasupathy. Predictive head movement tracking using a Kalman filter. *IEEE Transactions on Systems, Man and Cybernetics – Part B: Cybernetics*, 27(2):326–331, April 1997.

[43] R. Koenen ed. Overview of the mpeg-4 standard. http://drogo.cselt.it/mpeg/standards/mpeg-4/mpeg-4.htm.

[44] B. W. Leach. An introduction to Kalman filtering. Technical report, National Aeronautical Establishment, March 1984.

[45] S. Li, M. Chiba, and S. Tsuji. Estimating camera motion precisely from omni-directional images. In *IROS '94*, pages 1126–1132, 1994.

[46] J. Liang, C. Shaw, and M. Green. On temporal-spatial realism in the virtual reality environment. In *UIST'91*, 1991.

[47] J. E. Lloyd, J. S. Beis, D. K. Pai, and D. G. Lowe. Model-based telerobotics with vision. In *Proc. of 1997 IEEE Int. Conf. on Robotics and Automation*, pages 1297–1304, Albuquerque, New Mexico, 1997.

[48] M. M Mallem, F. Chavand, and E. Colle. Computer-assisted visual perception in teleoperated robotics. *Robotica*, 10:93–103, 1992.

[49] D. Marr and E. Hildreth. Theory of edge detection. *Proc. of Royal Society of London B*, 207:198–217, 1980.

[50] J. Mulligan. Fast calibrated stereo vision for manipulation. In *Proc. of 1996 IEEE Int. Conf. on Robotics and Automation*, pages 2326–2331, Minneapolis, Minnesota, April 1996.

[51] M. C. Nechyba and Y. Xu. SM^2 for new space station structure: Autonomous locomotion and teleoperation control. In *Proc. of 1994 IEEE Int. Conf. on Robotics and Automation*, volume 2, pages 1765–1770, 1994.

[52] P. Newman and H. Durrant-Whyte. Using sonar in terrain-aided underwater navigation. In *Proc. of 1998 IEEE Int. Conf. on Robotics and Automation*, pages 440–445, Leuven, Belgium, May 1998.

[53] K. Nickels and S. Hutchinson. Weighting observations: The use of kinematic models in object tracking. In *Proc. of 1998 IEEE Int. Conf. on Robotics and Automation*, pages 1677–1682, Leuven, Belgium, May 1998.

[54] E. Oyama, N. Tsunemoto, S. Tachi, and Y. Inoue. Experimental study on remote manipulation using virtual reality. *Presence*, 2(2):112–124, 1993.

[55] J. R. Parker. *Practical Computer Vision Using C.* John Wiley & Sons, New York, 1994.

[56] C. Pégard and E. M. Mouaddib. A mobile robot using a panoramic view. In *Proc. of 1996 IEEE Int. Conf. on Robotics and Automation*, pages 89–94, Minneapolis, Minnesota, April 1996.

[57] M. Petrou and J. Kittler. Optimal edge detectors for ramp edges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(5):483–491, 1991.

[58] Proxim, Inc., Mtn. View, CA. *ProxLink Radio Module User's Manual*, revision 4.2 edition, November 16 1993.

[59] A. Puri, H.-M. Hang, and D. L. Schilling. An efficient block-matching algorithm for motion-compensated coding. In *Proc. of ICASSP*, pages 25.5.1–25.4.4, 1987.

[60] S. I. Roumeliotis, G. S. Sukhatme, and G. A. Bekey. Smoother based 3d attitude estimation for mobile robot localization. In *Proc. of 1999 IEEE Int. Conf. on Robotics and Automation*, pages 1979–1986, Detroit, Michigan, May 1999.

[61] A. P. Sage. *Optimum Systems Control.* Prentice-Hall, 1968.

[62] A. P. Sage and J. L. Melsa. *Estimation Theory with Applications to Communications and Control.* McGraw-Hill, 1971.

[63] A. Said and W. Pearlman. A new, fast, and efficient image codec based on set partitioning in hierarchical trees. *IEEE Trans. on Circuits and Systems for Video Technology*, 6:243–250, 1996.

[64] C. P. Sayers. Intelligent image fragmentation for teleoperation over delayed low-bandwidth links. In *Proc. of 1996 IEEE Int. Conf. on Robotics and Automation*, volume 2, pages 1363–1368, Minneapolis, Minnesota, 1996.

[65] C. P. Sayers, A. Lai, and R. P. Paul. Visual imagery for subsea teleprogramming. In *Proc. of 1995 IEEE Int. Conf. on Robotics and Automation*, volume 2, pages 1567–1572, 1995.

[66] S. Scheding, E. Nebot, and H. Durrant-Whyte. The detection of faults in navigation systems: a frequency domain approach. In *Proc. of 1998 IEEE Int. Conf. on Robotics and Automation*, pages 2217–2222, Leuven, Belgium, May 1998.

[67] D. Schilling and P. Cosman. Edge-enhanced image coding for low bit rates. In *Proc. 1999 Int. Conf. on Image Processing*, volume 3, pages 747–751, Kobe, Japan, October 1999.

[68] Robert Shaffer. Transmission of H.263 Video Over ATM Networks. Master's thesis, University of Alberta, 1999.

[69] J. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Trans. on Signal Processing*, 41:3445–3462, 1993.

[70] T. B. Sheridan. *Telerobotics, Automation, and Human Supervisory Control.* MIT Press, 1992.

[71] D. Southwell, A. Basu, M. Fiala, and J. Reyda. Panoramic stereo. In *Proceedings of ICPR '96*, pages 378–382, 1996.

[72] M. Suk and S.-M. Chung. A new image segmentation technique based on partition mode test. *Pattern Recognition*, 16(5):469–480, 1983.

[73] S. Tachi and K. Yasuda. Evaluation experiments of a teleexistence manipulation system. *Presence*, 3(1):35–44, 1994.

[74] K. Taylor and B. Dalton. Issues in internet telerobotics. In *Proc. of Int. Conf. on Field and Service Robotics*, pages 151–157, Canberra, Australia, 1997.

[75] K. Taylor and J. Trevelyan. Australia's telerobot on the web. http://telerobot.mech.uwa.edu.au/robot/singapor.htm.

[76] S. Wei, Y. Yagi, and M. Yachida. Building local floor map by use of ultrasonic and omni-directional vision sensor. In *Proc. of 1998 IEEE Int. Conf. on Robotics and Automation*, Leuven, Belgium, May 1998.

[77] K. J. Wiebe and A. Basu. Modelling ecologically specialized biological visual systems. *Pattern Recognition*, 30(10):1687–1703, 1997.

[78] M. Willebeek-LeMair and Z. Shae. Robust h.263 video coding for transmission over the internet. In *Infocom '98*, pages 225–232, San Francisco, California, 1998.